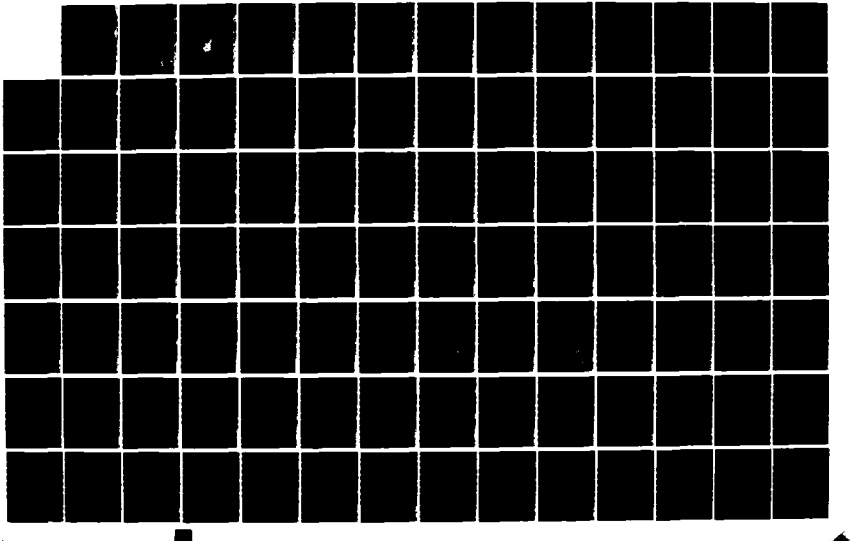
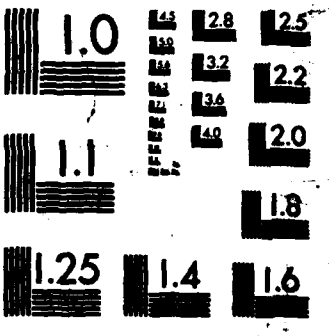
AD-A165 029 METHODOLOGY INVESTIGATION PROGRAM FLOW ANALYZER VOLUME 2(U) ARMY ELECTRONIC PROVING GROUND FORT HUACHUCA AZ E L ANDERSON SEP 85 1/2

UNCLASSIFIED F/G 9/2 NL





UNCLASSIFIED

AD No.
TECOM Project No. 7-CO-PB4-EP1-001

METHODOLOGY INVESTIGATION

FINAL REPORT

PROGRAM FLOW ANALYZER

VOLUME II

BY

EDWARD L. ANDERSON

US ARMY ELECTRONIC PROVING GROUND
Fort Huachuca, Arizona 85613-7110

SEPTEMBER 1985

DISTRIBUTION UNLIMITED

US ARMY TEST & EVALUATION COMMAND
Aberdeen Proving Ground, MD 21005-5055

UNCLASSIFIED

Document Number: UM-01-05

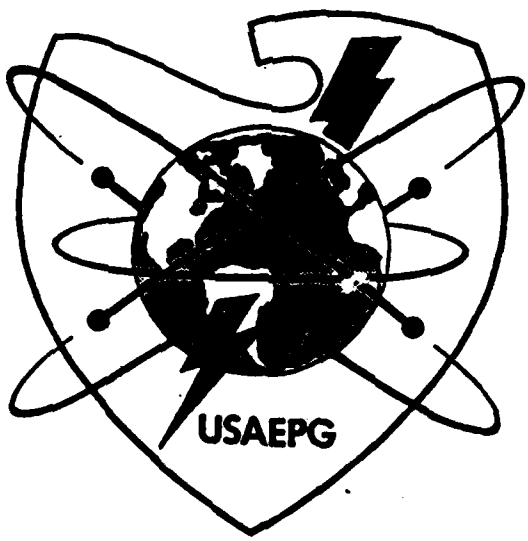
July 1985

ACAP

TABLE-DRIVEN

ASSEMBLY LANGUAGE CODE ANALYSIS PROGRAM

TECHNICAL USER MANUAL



U.S. ARMY ELECTRONIC PROVING GROUND

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

1.1 PURPOSE

The table-driven Assembly Code Analysis Program (ACAP) is one component in a family of automated software assessment tools. These tools extract information about software quality features of the target software being examined. ACAP provides a translator which can be modified to tailor it to the specific assembly language of the target software. Provision is also made in the ACAP system for generating reports from information collected by other language translators.

1.2 SCOPE

This document is intended to be an informal technical user manual to aid the analyst performing the software assessment. It is assumed that the user is familiar with the host computer system (VAX/VMS) utilities, particularly the EDT editor, and the FORTRAN language. An understanding of static software assessment techniques, computer languages, and the target system software is essential.

1.3 DESCRIPTION

The ACAP system has been developed on the Digital Equipment VAX series of computers utilizing VMS. The ACAP is written entirely in VAX/DCL and FORTRAN. The table driven ACAP system accepts input defining the mnemonics of the assembler language to be analyzed. Depending on the particular language, the statement parser may need modification and subsequent linkage within the code translation section of ACAP. ACAP determines information about the structure, complexity, use of instructions and variables, and selected quality parameters.

A general purpose report writer interfaces with ACAP and other code analysis programs. Data generated by these programs are provided via files to the Report Writer. These files supply the data that creates the software assessment reports, and in addition, are concatenated into a single master file.

The ACAP system also provides the capability to create a paginated, sequenced listing to assist in software assessment activities.

The users manual

~~The following sections~~ describes the usage of ACAP, define the interfaces among each of the subsystems, describe the files used within ACAP, and illustrate each of the reports created.

2.0 ACAP USAGE

The ACAP program is run by typing "@ACAP". A main menu (Figure 2-1) is then displayed. The main menu is a list of actions/functions available to the user. Each of the selections on the main menu is described in the following sections. Every menu of ACAP has a help file which explains the usage and content of the menu.

```
                        ACAP MAIN MENU

DISPLAY             Display ACAP Capabilities . . . . . 1
INSTRUMENT CODE     Instrument Source Code . . . . . 2
S/W ASSESSMENT      Perform Software Assessment . . . . . 3
REPORT WRITER       Run ACAP Report Writer . . . . . 4
VIEW/PRINT          View/Print Reports . . . . . 5
NEW LANGUAGE        Create New Language Capability . . . . 6
HELP                Display Helpful Information . . . . . H
EXIT TO VMS         Terminate ACAP and Exit to VMS . . . . E
What would you care to do now?:
```

FIGURE 2-1 - ACAP MAIN MENU

2.1 DISPLAY ACAP CAPABILITIES

This selection assists the user in understanding the capabilities of ACAP. The user is presented with the menu shown in Figure 2-2 from which to select the information to be displayed.

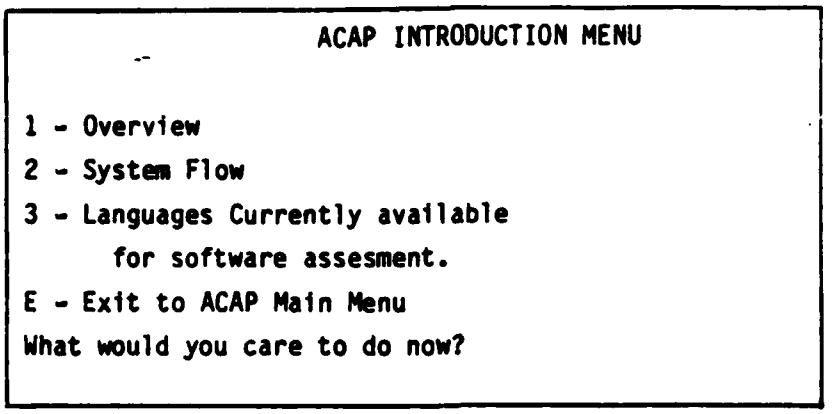


FIGURE 2-2 - ACAP INTRODUCTION MENU

2.2 INSTRUMENT SOURCE CODE

This selection assists the user in instrumenting source code files for subsequent processing by ACAP. Instrumentation is necessary since there are inconsistencies in describing how or where a procedure might begin and is accomplished by inserting a line of information into the source file to define segments of code for ACAP to assess and generate reports. This eliminates ambiguities and assures that the code is processed consistently for static analysis. Instrumented lines are not regarded as source by ACAP. This is explained further in Appendix B. The Instrument Code Menu is shown in Figure 2-3. A description of each of the numbered selections in this menu is presented in the following paragraphs:

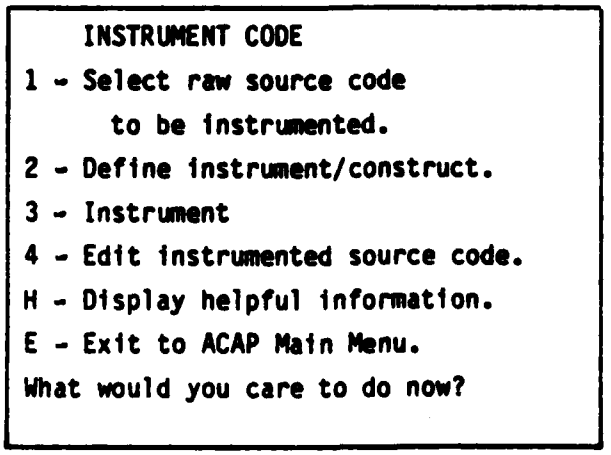


FIGURE 2-3 - INSTRUMENT CODE MENU

2.2.1 Select Raw Source Code to be Instrumented

This selection queries the user for the name of the source file to be instrumented. The name must be a legal VMS file name including extension. The specified file name becomes the input file name used by menu item 3.

The specified file may reside in another directory in which case the Directory Path should be entered with the File Name. If this approach is used the user should be certain they have appropriate access to the specified directory. (Note: The combined Directory Path and File Name cannot exceed 31 characters.)

2.2.2 Define Instrument/Construct

This selection allows the user to define the constructs to be used in the source code instrumentation. Upon selection of this menu item, the user is placed into the VAX editor with a template of a namelist file displayed for editing. After editing, this file is used to assist in instrumenting the source code when the user selects menu item 3.

The template file IRM.MTL contains comments to assist the user in understanding the instrumentation process. Below is the contents of this file.

```
$IRM
```

The instrument value is the string which the code translator recognizes as the start of a piece of instrumentation. It should not be recognizable by the compiler, and it has a maximum length of 15 characters. An example instrument value is "C**".

```
INSTRUMENT_VALUE = ' ',
```

Code constructs are pieces of source code which hint at the correct place to put instrumentation. For example in FORTRAN, PROGRAM is a likely place for the start of either a module or a procedure. SUBROUTINE is a likely place for the start of a procedure. The program which aids in the instrumentation process searches for the code constructs and inserts a line containing the INSTRUMENT_VALUE in front of the line on which the code construct was found. If CODE_CONSTRUCT(N) is given the value ' ', it will be ignored.

```
CODE_CONSTRUCT1 = ' ',
CODE_CONSTRUCT2 = ' ',
CODE_CONSTRUCT3 = ' ',

$END
```

The user is expected to edit the values as appropriate and then exit from the editor in the normal manner which returns control to ACAP.

2.2.3 Instrument

This selection assists the user in doing the actual source code instrumentation. The user is prompted to specify the output file name by the query 'WHAT WOULD YOU LIKE TO CALL THE INSTRUMENTED SOURCE?'. The name must be a legal VMS file name including extension. No Directory Path should be entered here, the file must reside in the default directory. After the output file name is entered, the input file specified using menu item 1 is opened, and each line in the source file is searched for the character strings specified as CODE_CONSTRUCTS above. If a CODE_CONSTRUCT is found, a single line containing the INSTRUMENT_VALUE is written to the output file followed by

the line of source; otherwise, just the source line is written to the output file.

2.2.4 Edit Instrumented Source File

This selection allows the user to complete the instrumentation process started above. The user is placed into the VAX editor in order to edit the file output above. The user is expected to utilize the search function of the editor to locate the automatically instrumented lines containing the INSTRUMENT_VALUE and complete each of these lines as explained in Appendix B of this document. Exiting from the editor in the normal manner returns control to ACAP.

2.3 SOFTWARE ASSESSMENT

Selecting this item allows the user to perform software assessment on an instrumented source program. The menu shown in Figure 2-4 is displayed to the user from which the user is required to select the appropriate language.

```
                - A C A P - SOFTWARE ASSESSMENT
     ASSEMBLY LANGUAGE CAPABILITIES
**************************************************
 1 - 6800          6 -            11 -
 2 - M68000        7 -            12 -
 3 - SKC           8 -            13 -
 4 - PDP11         9 -            14 -
 5 -              10 -            15 -
**************************************************
 If the Language you want is not available, Press RETURN key.
 Please Select Assembly Language for ACAP S/W Assessment
```

FIGURE 2-4 - ACAP ASSEMBLY LANGUAGES

A message is displayed if the language is not currently available.

A description of each of the numbered selections on this menu is presented in the following paragraphs. After selection of a language the menu shown in figure 2.5 is displayed:

```
    SOFTWARE ASSESSMENT MENU

1 - Select Instrumented Source Code
      to be Processed by ACAP
2 - Process Code and Create Master File
H - Display Helpful Information
E - Exit to ACAP Main Menu
What would you care to do now?
```

FIGURE 2-5 - SOFTWARE ASSESSMENT MENU

2.3.1 Select Instrumented Source Code to be Processed by ACAP

This selection queries the user for the file name of the instrumented source code as follows:

```
Select the Instrumented Source Code
Enter the Filename
(Note:  The file must reside in the default directory.):
Enter the Name of the Program in That File
(Must be a Legal VMS Filename Without Extension):
```

The Filename is used to locate the instrumented source which must reside in the default directory. A blank response to this inquiry is not allowed and will cause the prompt to reoccur.

The program name is used to name report files output by the Report Writer.

The user is then queried for the classification of the program source as follows:

```
      Legal Classifications
        'U' = Unclassified
        'C' = Confidential
        'S' = Secret
        'T' = Top Secret

Classification  of this program:
```

If the user enters something other than one of the legal classifications, the message "That is not a legal classification. Please try again." will be displayed.

2.3.2 Process Code and Create Master File

This selection processes the instrumented source code in the file specified by menu item 2 through the Code Translator and the Static Analyzer automatically generating all the data necessary for all the reports. After completion of the translation and static analysis, the user is prompted to enter a name for a Master File to contain all the output data for the reports. It is recommended that the Master File name has an extension, such as MAS, that follows a user-defined convention to allow easy identification of Master Files.

2.4 REPORT WRITER

This selection allows the user to select specific ACAP reports to be generated. The user is first asked "WHAT IS THE NAME OF THE MASTER FILE?:". The name must be a legal VMS file name including extension. After specifying the file name, the menu shown in Figure 2-6 is displayed to the user.

```
                    CREATE REPORTS

     1.  Module Report
     2.  Summary and Compliance Reports
     3.  Software Quality Metrics Report
     4.  Calls Report
     5.  Structure Diagrams
     6.  Undefined Externals Report
     7.  Variable Usage Report
     8.  Sequenced/Paginated Listing
     9.  Macro Usage Report
Select the reports to generate (e = quit, h = help, r = run):
```

FIGURE 2-6 - ACAP REPORT GENERATION MENU

To select a report, the user types the report number, as shown on the menu, followed by a RETURN. The selected report title is then highlighted using inverse video. Once selected, a report can be deselected by again typing the report number followed by a RETURN and the report title will no longer be high lighted. Any number of reports can be selected for generation.

After the user has selected all the reports he desires, the user types 'R <RETURN>' to have the reports run.

To obtain help, the user types 'H <RETURN>'.

To exit report generation and return to the main menu, the user types 'E <RETURN>'.

A sample of all the reports is included in Section 6. Note that the information in the reports depends on the information in the master file. Not all data will or can be extracted by the Code Translator/Static Analyzer for all languages. For information on the metrics that are being extracted for a particular language, see the appropriate appendix.

2.4.1 Module Report

The Module Report (in a file named "Program_name.MOD") lists the module names, the number of procedures in each module, the procedure names and the page number of the procedure in the sequenced paginated listing.

2.4.2 Summary and Compliance Reports

The Summary and Compliance Reports (in a file named "Program_name.CPL) indicate the degree of compliance that the software being analyzed has with selected metrics. The Summary Report lists the number of compliant and non-compliant procedures, the percent of compliant and non-compliant procedures, the total value, and the average value for each metric. The Compliance Report consists of two tables. The first is the Numerical Compliance Table which lists the value of each metric for each procedure. The second is the Compliance Table which indicates whether each procedure is compliant with the compliance values.

After selection, the following question is displayed to the user: WOULD YOU LIKE TO CHANGE THE COMPLIANCE VALUES? If the user enters 'Y', he will be able to change compliance values as discussed in Section 2.4.2.1; otherwise, the user is prompted to select the metrics for the Summary and Compliance Reports using the menu shown in Figure 2-7.

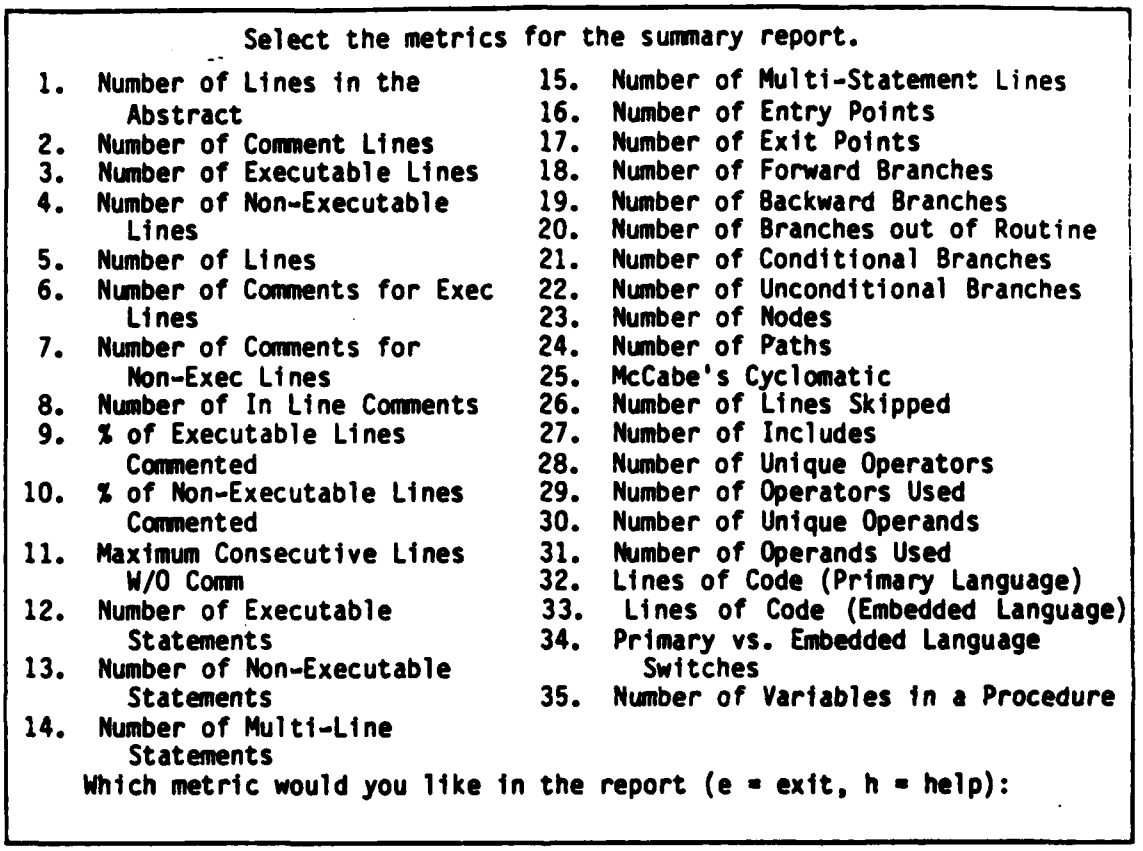


FIGURE 2-7 - METRIC SELECTION

To select a metric, the user types the metric number, as shown on the menu, followed by a RETURN. The selected metric is then highlighted using inverse video. Once selected a metric can be deselected by again typing the metric number followed by a RETURN. Only ten (10) metrics may be selected at a time. Error messages will appear if the selection of an eleventh metric is attempted or if an attempt is made to select a metric which has been determined to be invalid for the selected language.

To obtain help, the user types 'H <RETURN>'.

To terminate selection of the metrics and return to the main menu, the user types 'E <RETURN>'.

2.4.2.1 Changing Compliance Values

The user is placed in the EDT editor in keypad mode and allowed to edit the REPORT.NML file. This is a Namelist file used in creating the Compliance and Summary Reports. The following is displayed to assist the user in the editing process:

```
$SUM_REP_CONFIG
!How to Change the Wording of the Report Headings and the Compliance Values
!
!The report heading for each metric consists of the two lines:  COL_HD_LN1
!and COL_HD_LN2.  To change the heading find the array entry which corresponds
!to the desired metric, and edit the headings.  Do not change the total number
!of characters or the report will not look right.

!The compliance value is stored in COMPLY_VAL.  To change the numerical value
!find the correct array entry and change it.

!The LT GT FLAG is tested to determine if the value of the metric must be <=
!or >= the compliance value.  If LT_GT_FLAG is given a value of T or TRUE,
!then the metric must be greater than or equal to the compliance value.  If
!it is given a value of F or FALSE then the metric must be less than or equal
!to the compliance value.
!
!The totals of some of the metrics are given in the summary report.  If the
!variable SHOW_TOTAL is TRUE then the total is shown.  If it is FALSE then the
!total is not shown.
!       Please look for the appropriate entries below:
! Number of lines in the abstract
COL_HD_LN1(1) = 'ABSTRACT|',
COL_HD_LN2(1) = 'LINES   |',
COMPLY_VAL(1) = 3,
LT_GT_FLAG(1) = T,
SHOW_TOTAL(1) = T,
```

```
! Number of comment lines
COL_HD_LN1(2) = 'COMMENT |',
COL_HD_LN2(2) = ' LINES  |',
COMPLY_VAL(2) = 100,
LT_GT_FLAG(2) = F,
SHOW TOTAL(2) = T,
! Number of executable lines
COL_HD_LN1(3) = ' EXECU |',
COL_HD_LN2(3) = ' LINES  |',
COMPLY_VAL(3) = 100,
LT_GT_FLAG(3) = F,
SHOW_TOTAL(3) = T,
! Number of non-executable lines
COL_HD_LN1(4) = 'NON-EXEC|',
COL_HD_LN2(4) = ' LINES  |',
COMPLY_VAL(4) = 100,
LT_GT_FLAG(4) = F,
SHOW_TOTAL(4) = T,
! Total number of lines
COL_HD_LN1(5) = ' TOTAL  |',
COL_HD_LN2(5) = ' LINES  |',
COMPLY_VAL(5) = 100,
LT_GT_FLAG(5) = F,
SHOW_TOTAL(5) = T,
! Number of comments in exec. stmts.
COL_HD_LN1(6) = 'COMMENTS|',
COL_HD_LN2(6) = '(EXECU.)|',
COMPLY_VAL(6) = 200,
LT_GT_FLAG(6) = F,
SHOW_TOTAL(6) = T,
! Number of comm. in non-exec. stmts.
COL_HD_LN1(7) = 'COMMENTS|',
COL_HD_LN2(7) = '(NON-EX)|',
COMPLY_VAL(7) = 50,
LT_GT_FLAG(7) = T,
SHOW_TOTAL(7) = T,
```

```
! Number of inline comments
COL_HD_LN1(8) = 'IN-LINE |',
COL_HD_LN2(8) = 'COMMENTS|',
COMPLY_VAL(8) = 100,
LT_GT_FLAG(8) = F,
SHOW_TOTAL(8) = T,
! % of executable line commented
COL_HD_LN1(9) = '% COMM. |',
COL_HD_LN2(9) = ' EXEC.  |',
COMPLY_VAL(9) = 100,
LT_GT_FLAG(9) = T,
SHOW_TOTAL(9) = F,
! % of non-executable line commented
COL_HD_LN1(10) = '% COMM. |',
COL_HD_LN2(10) = ' NON-EX |',
COMPLY_VAL(10) = 100,
LT_GT_FLAG(10) = T,
SHOW_TOTAL(10) = F,
! Number of consecutive lines w/o comm.
COL_HD_LN1(11) = 'MAX W/O |',
COL_HD_LN2(11) = 'COMMENT |',
COMPLY_VAL(11) = 4,
LT_GT_FLAG(11) = F,
SHOW_TOTAL(11) = F,
! Number of executable statements
COL_HD_LN1(12) = ' EXECU. |',
COL_HD_LN2(12) = ' STMTS. |',
COMPLY_VAL(12) = 100,
LT_GT_FLAG(12) = F,
SHOW_TOTAL(12) = T,
! Number of non-executable statements
COL_HD_LN1(13) = ' NON-EX |',
COL_HD_LN2(13) = ' STMTS. |',
COMPLY_VAL(13) = 50,
LT_GT_FLAG(13) = F,
SHOW_TOTAL(13) = T,
```

```
! Number of multi-line statements
COL_HD_LN1(14) = 'MLT-LINE|',
COL_HD_LN2(14) = ' STMTS. |',
COMPLY_VAL(14) = 0,
LT_GT_FLAG(14) = F,
SHOW_TOTAL(14) = T,
! Number of multi-statement lines
COL_HD_LN1(15) = 'MLT-STMT|',
COL_HD_LN2(15) = ' LINES  |',
COMPLY_VAL(15) = 0,
LT_GT_FLAG(15) = F,
SHOW_TOTAL(15) = T,
! Number of entry points
COL_HD_LN1(16) = ' ENTRY  |',
COL_HD_LN2(16) = 'POINTS  |',
COMPLY_VAL(16) = 1,
LT_GT_FLAG(16) = F,
SHOW_TOTAL(16) = T,
! Number of exit points
COL_HD_LN1(17) = '  EXIT  |',
COL_HD_LN2(17) = ' POINTS |',
COMPLY_VAL(17) = 1,
LT_GT_FLAG(17) = F,
SHOW_TOTAL(17) = T,
! Number of forward branches
COL_HD_LN1(18) = 'FORWARD |',
COL_HD_LN2(18) = 'BRANCHES|',
COMPLY_VAL(18) = 0,
LT_GT_FLAG(18) = F,
SHOW_TOTAL(18) = T,
! Number of backward branches
COL_HD_LN1(19) = 'BACKWARD|',
COL_HD_LN2(19) = 'BRANCHES|',
COMPLY_VAL(19) = 0,
LT_GT_FLAG(19) = T,
SHOW_TOTAL(19) = T,
```

```
! Number of branches out of routine
COL_HD_LN1(20) = 'OUTWARD |',
COL_HD_LN2(20) = 'BRANCHES|',
COMPLY_VAL(20) = 0,
LT_GT_FLAG(20) = F,
SHOW_TOTAL(20) = T,
! Number of conditional branches
COL_HD_LN1(21) = '  COND. |',
COL_HD_LN2(21) = 'BRANCHES|',
COMPLY_VAL(21) = 0,
LT_GT_FLAG(21) = F,
SHOW_TOTAL(21) = T,
! Number of unconditional branches
COL_HD_LN1(22) = ' UNCOND.  |',
COL_HD_LN2(22) = ' BRANCHES |',
COMPLY_VAL(22) = 0,
LT_GT_FLAG(22) = F,
SHOW_TOTAL(22) = T,
! Number of nodes
COL_HD_LN1(23) = '  NODES   |',
COL_HD_LN2(23) = '          |',
COMPLY_VAL(23) = 0,
LT_GT_FLAG(23) = F,
SHOW_TOTAL(23) = T,
! Number of paths
COL_HD_LN1(24) = '  PATHS   |',
COL_HD_LN2(24) = '          |',
COMPLY_VAL(24) = 0,
LT_GT_FLAG(24) = F,
SHOW_TOTAL(24) = T,
! McCabe's cyclomatic
COL_HD_LN1(25) = 'McCABES |',
COL_HD_LN2(25) = 'COMPLEX.|',
COMPLY_VAL(25) = 10,
LT_GT_FLAG(25) = F,
SHOW_TOTAL(25) = F,
```

```
! Number of lines skipped
COL_HD_LN1(26) = ' LINES |',
COL_HD_LN2(26) = 'SKIPPED|',
COMPLY_VAL(26) = 0,
LT_GT_FLAG(26) = F,
SHOW_TOTAL(26) = T,
! Number of includes
COL_HD_LN1(27) = 'INCLUDES|',
COL_HD_LN2(27) = '        |',
COMPLY_VAL(27) = 0,
LT_GT_FLAG(27) = F,
SHOW_TOTAL(27) = T,
! Number of unique operators
COL_HD_LN1(28) = ' UNIQUE |',
COL_HD_LN2(28) = 'OPERAT. |',
COMPLY_VAL(28) = 0,
LT_GT_FLAG(28) = F,
SHOW_TOTAL(28) = T,
! Number of operators used
COL_HD_LN1(29) = 'OPERAT. |',
COL_HD_LN2(29) = '  USED  |',
COMPLY_VAL(29) = 0,
LT_GT_FLAG(29) = F,
SHOW_TOTAL(29) = T,
! Number of unique operands
COL_HD_LN1(30) = ' UNIQUE |',
COL_HD_LN2(30) = 'OPERANDS|',
COMPLY_VAL(30) = 0,
LT_GT_FLAG(30) = F,
SHOW_TOTAL(30) = T,
! Number of operands used
COL_HD_LN1(31) = ' OPERANDS |',
COL_HD_LN2(31) = '   USED   |',
COMPLY_VAL(31) = 0,
LT_GT_FLAG(31) = F,
SHOW_TOTAL(31) = T,
```

```
! Number of lines of primary code
COL_HD_LN1(32) = 'LINES OF|',
COL_HD_LN2(32) = 'PRIMARY |',
COMPLY_VAL(32) = 0,
LT_GT_FLAG(32) = F,
SHOW_TOTAL(32) = T,
! Number of lines of embedded code
COL_HD_LN1(33) = 'LINES OF|',
COL_HD_LN2(33) = 'EMBEDDED|',
COMPLY_VAL(33) = 0,
LT_GT_FLAG(33) = F,
SHOW_TOTAL(33) = T,
! Number of switches: Primary vs. Embedded
COL_HD_LN1(34) = 'SWITCHES |',
COL_HD_LN2(34) = 'PRI/EMBD |',
COMPLY_VAL(34) = 0,
LT_GT_FLAG(34) = F,
SHOW_TOTAL(34) = T,
! Number of variables in a procedure
COL_HD_LN1(35) = 'NUM. OF |',
COL_HD_LN2(35) = ' VARS.  |',
COMPLY_VAL(35) = 50,
LT_GT_FLAG(35) = F,
SHOW_TOTAL(35) = T,
! The rest are not used.
COL_HD_LN1(36) = '          |',
COL_HD_LN2(36) = '          |',
COMPLY_VAL(36) = 0,
LT_GT_FLAG(36) = F,
SHOW_TOTAL(36) = T,
```

```
!
COL_HD_LN1(50) = '           |',
COL_HD_LN2(50) = '           |',
COMPLY_VAL(50) = 0,
LT_GT_FLAG(50) = F,
SHOW_TOTAL(50) = T,
YES      = '   YES     |',
NO       = '    NO     |',
BLANK    = '           |',
$END
```

Exiting from the editor in the normal manner returns control to ACAP. The user will then be able to select the metrics for the Summary and Compliance Reports as discussed previously.

2.4.3 Software Quality Metrics Report

The Software Quality Metrics Report (in file name "Program_name.SQM") itemizes all of the metrics listed in Figure 2-7 for each procedure. Not all metrics can be collected for all languages. If a metric has been determined to be "not applicable" for a given language an entry is made, by the user, in the "INIT_MET" Namelist which resides in the Parser Table. (See Appendix D.) The entry is given a value of -1 and the metric will not be processed further. "N/A" for "not applicable" will be printed on the report.

2.4.4 Calls Report

The Calls Report (in a file named "Programm_name.CAL") lists all the internal calls from each procedure within a module.

2.4.5 Structure Diagrams

The Structure Diagrams (in a file named "Program_name.STR") are generated for each procedure showing up to fourteen (14) levels of procedure calls.

2.4.6 Undefined Externals Report

The Undefined Externals Report (in a file named "Program_name.EXT") lists all the unresolved external references in each procedure.

2.4.7 Variable Usage Report

Variable Usage Reports (in a file named "Program_name.VUR") are generated for both local and global variables. For global variables, the report lists each global variable, the variable type (if applicable), the procedures in which it appears, and the number of times it is modified, referenced, tested, defined, or passed as an argument in the procedure. For each procedure, the report lists all variables, whether they are local, global or passed to the procedure as arguments, their variable type, and the number of times they are modified, referenced, tested, defined or passed as an argument in the procedure.

2.4.8 Sequenced/Paginated Listing

A Sequential Listing (in a file named "Program_name.LST") is generated for each procedure numbering each source line and printing a page number on each page. These page numbers are referenced on the Module Report.

2.4.9 Macro Usage Report

Macro Usage Report (in a file named "Program_name".MUR") lists all Macro Definitions and usage found in the instrumented source.

2.5 VIEW AND PRINT

This selection allows the user to either view or print reports generated by the Report Writer. The menu shown in Figure 2-9 is displayed to the user.

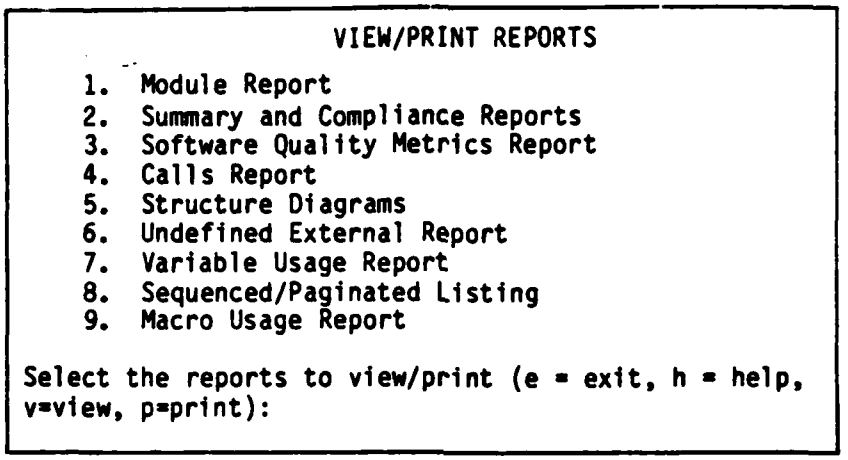


FIGURE 2-9 - VIEW/PRINT REPORT MENU

To select a report, the user types the report number, as shown on the menu, followed by a RETURN. The selected report title is then highlighted using inverse video. Once selected, a report can be deselected by again typing the report number followed by a RETURN, and the report title will no longer be highlighted. Any number of reports can be selected.

After selection is complete, the user types 'V (RETURN)' to view the reports or 'P(RETURN)' to print the reports.

When viewing reports, the user is placed in the VAX editor and so has all the editor capabilities available for viewing the selected reports. To aid the user in viewing reports a special function key has been established to set the screen to 132 character mode. The user simply types 'Gold' 'CTRL W' and 132 mode will be set. Upon exit the screen will automatically reset to 80 character mode.

Upon completion of the view or print process, the main view and print menu is displayed to the user for continuation of the selection process. To obtain help, the user types 'H(RETURN)'. To exit, the user types 'E(RETURN)'.

2.6 NEW LANGUAGE

This selection allows the user to develop the capability to process new languages for ACAP and to manually step through the software assessment process. The menu shown in Figure 2-8 is displayed to the user.

```
NEW LANGUAGE MENU

1 - Set up Parser Tables
2 - Select instrumented source code
      file to be processed.
3 - Run Code Translator
4 - Run Static Analyzer
5 - Create a Master File
H - Display helpful information.
E - Exit to ACAP Main Menu"
What would you care to do now?"
```

FIGURE 2-8 - NEW LANGUAGE MENU

2.6.1 Set-up Parser Tables

To set-up the parser table, the user is allowed to edit an existing parser table. (For an example of a Parser Table see Appendix D.) The user is placed in the VAX EDT editor in keypad mode and can use normal editing commands to create the desired new parser table. The following information is stored in the table:

Variable Name	Type	(Dimension)	Description
N_ASSM	Int 4		Number of assembly language mnemonics.
ASSM_INSTS	Char 8	(200)	Assembly language mnemonics.
ASSM_CODES	Int 4	(200)	Code for each mnemonic.
ASSM_MODES	Int 4	(200)	Address mode for each mnemonic.
ASSM_SPECS	Int 4	(200)	Special information about the instruction.
N_USER	Int 4		Number of user-defined instructions.
USER_INSTS	Char 8	(100)	User mnemonics.
USER_CODES	Int 4	(100)	Code for each user instruction.
USER_MODES	Int 4	(100)	Address mode for each user instruction.
USER SPECS	Int 4	(100)	Special information about each user instruction.

2.6.2 Select Instrumented Source Code File to be Processed

This selection queries the user for the filename of the instrumented source code as follows:

```
Select the Instrumented Source Code
Enter the Filename:
What is the name of the program in that file:
```

The program name allows different source files to be analyzed for a given program. ("Program_Name" is used for all report names.)

2.6.3 Run Code Translator

This selection runs the Code Translator. For more information on the Code Translator, see Section 3.

2.6.4 Run Static Analyzer

This selection runs the Static Analyzer. For more information on the Static Analyzer, see Section 3.

2.6.5 Create Master File

This selection creates one file containing all of the files from the ACAP run. The user is asked for the name of the Master File, and a check is made and a warning issued if the file already exists. The Master File is created from the ACAP output files from the most recently completed ACAP run. If the user does not specify an extension such as MAS, an extension of NML is appended to the user-supplied name. The Master File is described in Appendix A.

2.7 HELP

The user may obtain helpful information in either of two ways:

1. Entering 'H <RETURN>' displays text explaining all selections available on this menu.

2. Entering 'H' with ['1' or '2' or ...'E'] <RETURN> displays text associated only with the specified menu item.

2.8 TERMINATE ACAP AND EXIT TO VMS

This selection terminates ACAP execution and returns the user to the VMS system.

3.0 THE ACAP SYSTEM

The structure of the ACAP system is shown in Figure 3-1. The system consists of an executive and four functional modules:

- Input Processor
- Code Translator
- Static Analyzer
- Report Writer

The executive functions as an interface to the user of ACAP and as a controller of the ACAP system modules. These functions are performed by means of a main menu feature and command procedures. These procedures assist the user in selecting input files and options for subsequent processing. The interface between the modules is accomplished using the various data files shown in the figure. These files consist of either "Namelist" formatted data or sequential data.

3.1 ACAP INPUT PROCESSOR

The ACAP executive passes control to this module when the user selects main menu item 2, Instrument Code, as discussed in the previous section on ACAP usage.

An interface diagram for the Input Processor is shown in Figure 3-2.

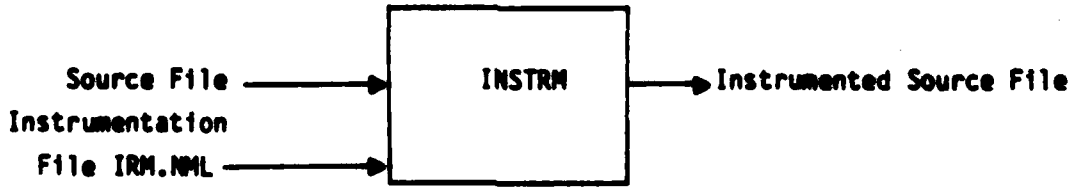


FIGURE 3-2 - INPUT PROCESSOR

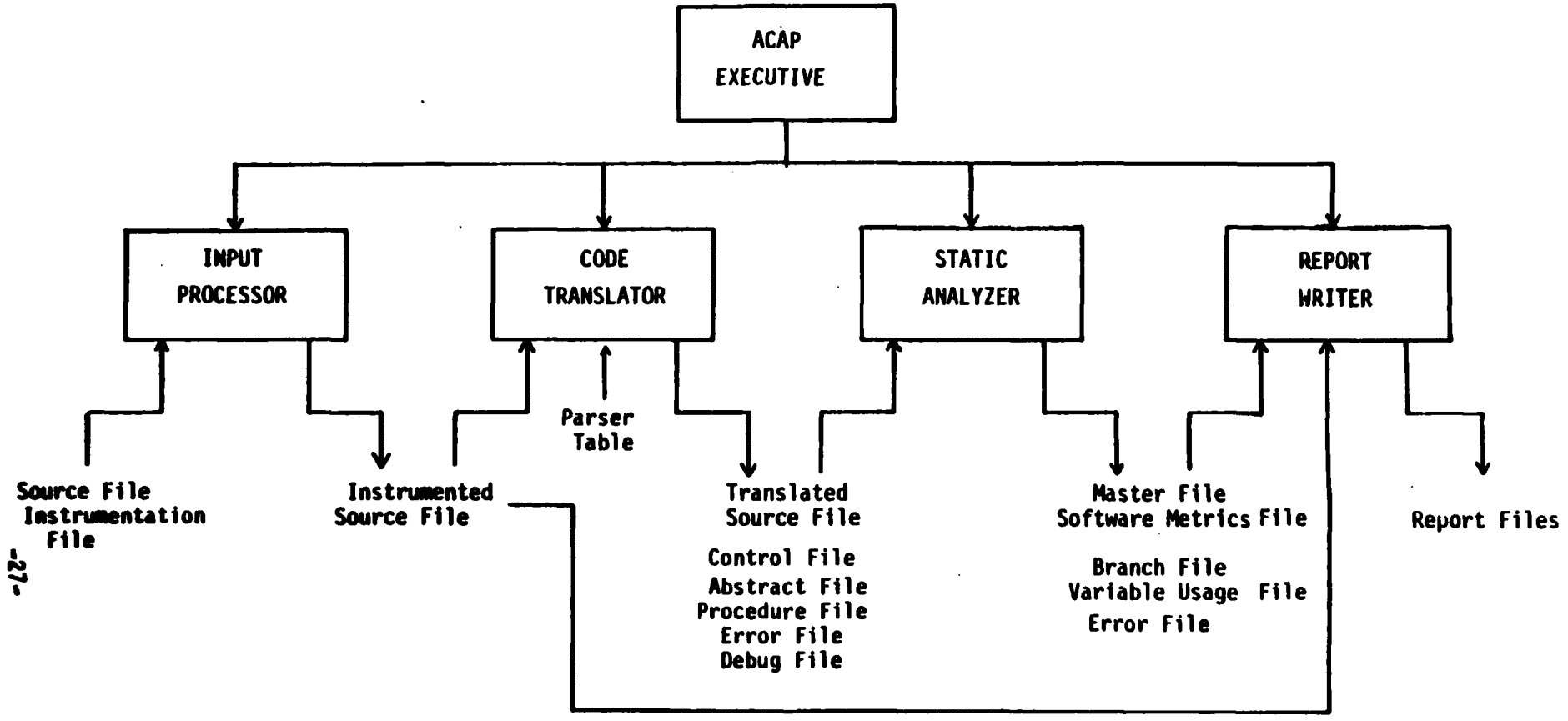


FIGURE 3-1 - THE STRUCTURE OF THE ACAP SYSTEM

3.1.1 Input

The Input Processor accepts the following input:

a. Source file - a file containing the assembly language source to be analyzed.

b. The instrumentation file (IRM.NML).

3.1.2. Processing

The Input Processor performs the function of semi-automatically instrumenting source files, which is further explained in Appendix B.

The Input Processor consists of the procedures shown in the structure charts of Section 4.0. INSTRM generates an instrumented source file by searching for user-defined constructs and inserting the specified instrumentation value, as defined in IRM.NML.

3.1.3 Outputs

The Input Processor generates the following output:

a. Instrumented Source File - a file containing the instrumented source code.

3.2 ACAP CODE TRANSLATOR

The code translator is entered via main menu item 3, Software Assessment.

3.2.1 Inputs

The Code Translator accepts the following inputs:

a. Instrumented Source File generated by the Input Processor.

b. PARSER.TBL - A file containing a description of the assembly language being translated.

c. Control.NML - A file containing identification information such as File Name, Source Language Selected, Date, Time, and Program Name.

3.2.2 Processing

The ACAP Code Translator consists of the procedures shown in the structure charts of Section 4. SCANNER reads each line of code from the instrumented source file and transforms these lines into a generic view of each statement which is independent of the specific language being processed. This generic information consists of codes or "tokens" which describe each statement including statement labels, label references, directives, parameters, and comments. The translation is controlled by a file (PARSER.TBL) containing tabular information about the operators and instructions of the source program's assembly language. This generic information is accumulated in a file (TRNSL.FIL) to be evaluated by the Static Analysis module. This file contains the necessary information about each procedure of the source code for subsequent static analysis. An interface diagram for the Code Translator is shown in Figure 3-3 below:

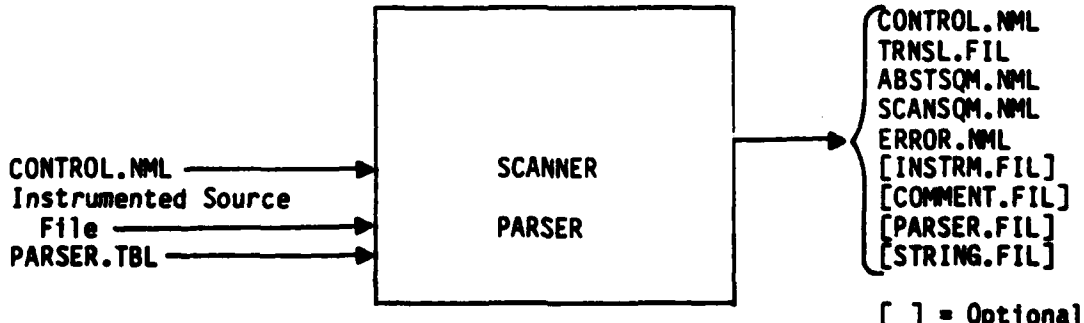


FIGURE 3-3 - CODE TRANSLATOR

3.2.3 Outputs

The Code Translator generates the following outputs:

a. Control File: CONTROL.NML - Scanner adds the Parser Table (Namelist PARSER_TABLE from file PARSER.TBL) to this file.

b. Translated Source File: TRNSL.FIL - A file containing the generic information on the assembly language being analyzed.

c. Abstract File: ABSTSQM.NML - A file containing data on the occurrence of instrumented abstracts.

d. Procedure File: SCANSQM.NML - A file containing the software metrics accumulated by the SCANNER for each procedure. See Section 5 for more information.

e. Error File: ERROR.NML - A file in which errors are accumulated regarding this analysis.

f. Debug File: INSTRM.FIL - An optional output file which contains debug information on instrumented lines controlled by INSTRM_PRNT in PARSER_TABLE.

g. Debug File: COMMENT.FIL - An optional output file which contains debug information on comments controlled by COMMENT_PRNT in PARSER_TABLE.

h. Debug File: PARSER.FIL - An optional output file which contains debug information on the parser's output for each line controlled by PARSER_PRNT in PARSER_TABLE.

i. Debug File: STRINGS.FIL - An optional output file which contains debug information on the strings found in each line controlled by STRINGS_PRNT in PARSER_TABLE.

3.3 ACAP STATIC ANALYZER

The ACAP executive passes control to this module when the user selects main menu items 3, Software Assessment.

3.3.1 Inputs

The Static Analyzer accepts the following inputs:

a. Control File: CONTROL.NML - A file used to control how the program is analyzed.

b. Translated Source File: TRNSL.FIL - A file generated by the Code Translator containing the translated assembly language source in generic form.

c. Abstract File: ABSTSQM.NML - A file containing data on the occurrence of instrumented abstracts.

d. Procedure File: SCANSQM.NML - A file generated by SCANNER containing certain software metrics.

e. Error File: ERROR.NML - A file in which errors are accumulated.

3.3.2 Processing

The Static Analyzer consists of the procedures shown in structure charts of Section 4. These procedures utilize the TRNSL.FIL created by the Code Translator to determine information about the software being assessed. Software quality metrics are calculated by the Static Analyzer as well as path and variable usage information. The results of this processing are written to various files that are utilized as input data to the Report Writer. An interface diagram for the Static Analyzer is shown in Figure 3-4 below:

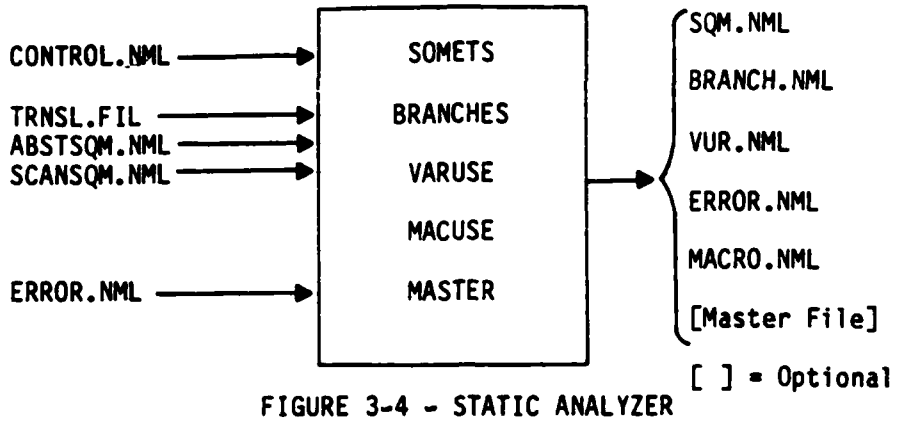


FIGURE 3-4 - STATIC ANALYZER

3.3.3 Outputs

The Static Analyzer generates the following outputs:

a. SQM.NML - A file containing the software quality metrics.

b. BRANCH.NML - A file containing label and branching information used in determining a programs structure and whether or not undefined interfaces exist.

c. MACRO.NML - A file containing MACRO usage data.

d. VUR.NML - A file containing the variable usage data.

e. ERROR.NML - A file containing any errors.

f. Master File - A user-specified file concatenating all of the static analysis output files.

3.4 ACAP REPORT WRITER

The ACAP executive passes control to this module when the user selects main menu item 4, Report Writer. In addition, the report writer was designed to be used by encoders and code analysis programs other than ACAP. Use of the Report Writer while outside ACAP is accomplished by issuing the command "@CAPWRITE". This command places the user directly into the Report

Writer's executive menu from which report selection and processing options are available. Either a single Master File or Individual Report Data Files may be utilized as input to the Report Writer. See Appendix A for detailed descriptions of these files.

3.4.1 Inputs

The Report Writer accepts the following inputs:

a. CONTROL.NML - A file used to control how the program is analyzed.

b. SQM.NML - A file containing the software quality metrics of the software being analyzed.

c. BRANCH.NML - A file containing the structure information on the software being analyzed.

d. VUR.NML - A file containing the variable usage information on the software being analyzed.

e. MACRO.NML - A file containing MACRO usage information on the software being analyzed.

f. Master File - A user-specified file which contains all of the static analysis output files concatenated into a single file.

g. Instrumented Source File - The instrumented assembly language source file.

3.4.2 Processing

The procedures composing the Report Writer are shown in the structure charts of Section 4. The interface diagram for the report writer is shown in Figure 3-5. These procedures read the master file and sort the data as necessary to create the output reports.

3.4.3 Outputs

The Report Writer generates the following outputs:

a. Module Report (Program_Name.MOD)
b. Summary Report (Program_Name.CPL)
c. Compliance Report (Program_Name.CPL)
d. Software Quality Metrics Report (Program_Name.SQM)
e. Calls Report (Program_Name.CAL)
f. Structure Diagrams (Program_Name.STR)
g. Undefined Externals Report (Program_Name.EXT)
h. Variable Usage Report (Program_Name.VUR)
i. Sequenced/Paginated Listing (Program_Name.LST)
j. MACRO Usage Report (Program_Name.MUR)

Examples of these reports are provided in Section 6.

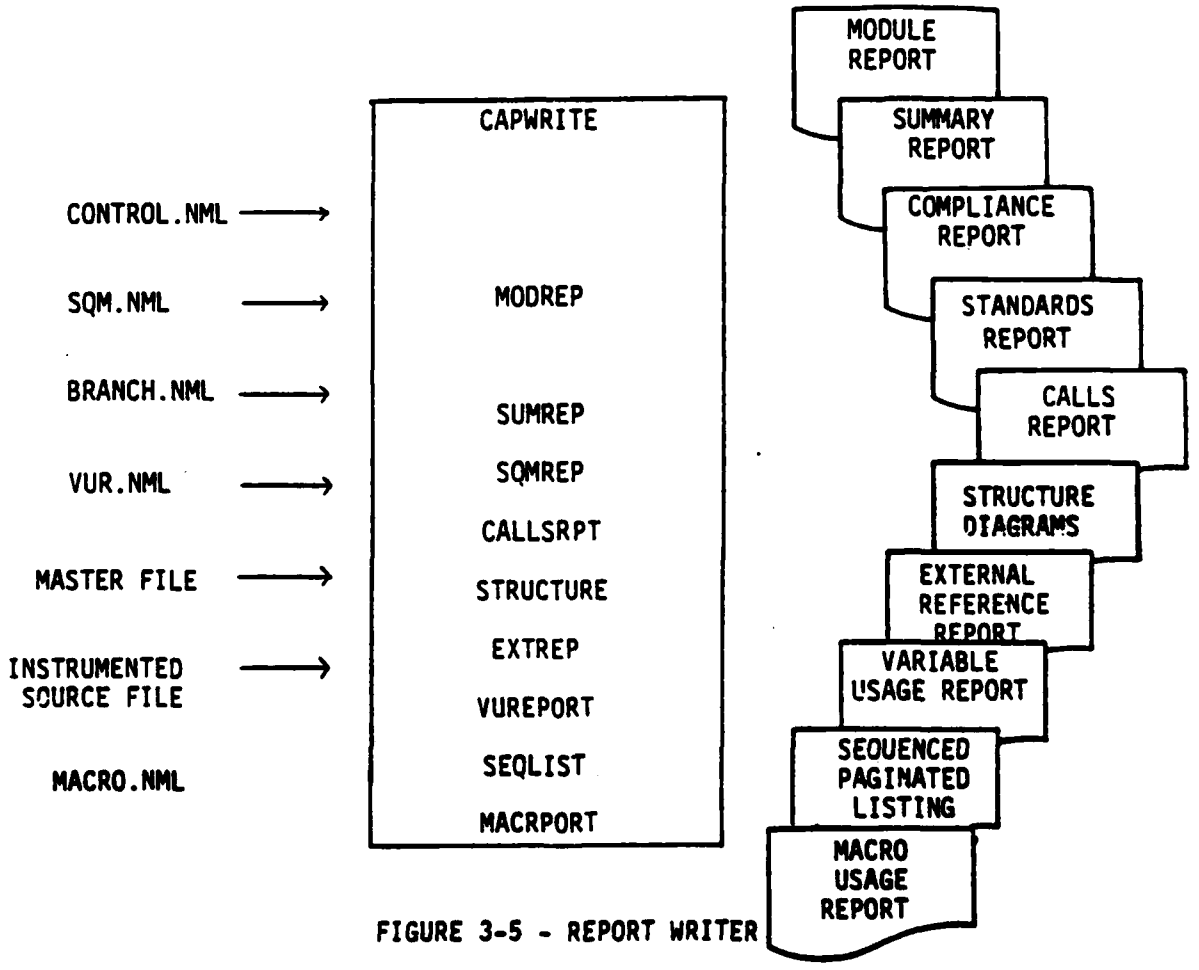


FIGURE 3-5 - REPORT WRITER

4.0 ACAP PROGRAM DESCRIPTION

The ACAP program is composed of the modules and procedures as shown in Figure 4-2. These figures contain structure charts for the entire program down to the subroutine level. In general, these charts only contain ACAP-specific procedures and not library procedures. A definition of the various structure chart constructs is shown in Figure 4-1.

The ACAP program is composed of modules and procedures written in VAX Fortran and DCL (DEC Command Language).

Table 4-1 contains an alphabetized list of the source files which compose the ACAP system. The following conventions have been used with respect the file name extensions:

BAN - DCL file used to display a banner to the user indicating what the program is doing.
COM - DCL files.
FOR - Fortran source files.
INC - Include files.
MEN - DCL files used to display a menu.
MTL - Master template files.
NML - Namelist files.

Table 4-2 shows which of these files must be linked together to form the various ACAP modules.

TABLE 4-1

ACAP PROGRAM FILES

Filename	Structure Chart Number	Description
6800.MEN	5.2.2	Banner for 6800 selection.
68000.MEN	5.2.3	Banner for 68000 selection.
ACAP.COM	0	Com file for running ACAP.
ACAPBAN.MEN	1	Banner for ACAP.
ACAPCAP.MEN	3.3.1	List of available languages.
ACAPEXIT.BAN	9	ACAP exit banner.
ACAPFLOW.MEN	3.2.1	ACAP flow diagram.
ACAPLAN.BAN	8	Banner used in New Language.
ACAPMAIN.MEN	2	ACAP main menu.
ASMCHOICE.COM	5.2	Com file to select Assembly Language.
ASMCHOICE.MEN	5.2.1	Menu of available languages.
ASSESS.BAN	5.1	Menu of software assessment options.
ASSESS.COM	5	Com file for running software assessment.
BINSEARCH.FOR	5.4.6.1	Subroutine BIN_SEARCH.
BLK2BLK.FOR	5.4.2.4	Subroutine for extracting next token.
BRANCHES.FOR	5.4.5	Main routine for obtaining branch metric data.
BRSQMNML.INC	-	Include file for BRCH_SQM namelist used in the file BRSQM.NML to communicate between BRANCHES and SQMETS.
CALLSNML.INC	-	Include file for CALLS Namelist.
CALLSRPT.FOR	6.2.3.1	Main routine for generating CALLS report.
CAPWRITE.COM	6.2	Com file for generating reports.

TABLE 4-1 (Continued)
ACAP PROGRAM FILES

Filename	Structure Chart Number	Description
CNTRLCOM.INC	-	Include file for SUM_REP and SUM_EXEC used to control metrics for summary report.
CNTRLREP.NML	-	Namelist for controlling summary report.
COMPLY.COM	6.2.1.2.2.1.1.1	Com file to edit REPORT.TPL (or MTL if non-existent).
CREPAR.COM	8	Com file used in New Language.
CREPAR1.COM	8	Com file used in New Language.
CREPATH.FOR	6.2.3.6.1.2	Subroutine CREATE_PATH.
CTEXEC.BAN	5.4.1	Banner indicating executing code translator.
CURSORPOS.FOR	6.2.1.2.1 6.2.1.2.2.1.3.1	Subroutine SET_CURSOR_POS.
DEFPRO.COM	5.4	Com file for doing translation and static analysis.
DRAWMENU.FOR	6.2.1.1 6.2.1.2.2.1.2	Subroutines DRAW_MENU and INVERT_PRINT.
ERROR.FOR	4.4.2.2	Subroutine PRINT_ERROR.
EXTREP.FOR	6.2.3.2	Main routine for generating External Reference Report.
GENDAT.COM	8	Com file used in New Language.
GENRPT.BAN	6.1	Banner for Report Writer.
GENRPT.COM	6	Com file for Report Writer.
GETCHR.FOR	6.2.1.3.1	Function GETCHR-single character synchronous read.
GIVEHELP.FOR	6.2.1.3 6.2.1.2.2.1.4	Subroutine GIVE_HELP for writing help files.

TABLE 4-1 (Continued)

ACAP PROGRAM FILES

Filename	Structure Chart Number	Description
HALSCOM.INC	-	Include file used by PAR68T.FOR and SCAN68T.FOR for passing HALSTEADS information.
HALSKC.INC	-	Include file used by PARSKC.FOR and SCANSKC.FOR for passing Halsteads information.
IDNML.INC	-	Include file for ID namelist.
INEDT.BAN	4.3.1 4.5.1	Banner indicating entry into editor.
INSEXEC.BAN	4.4.1	Banner for doing instrumentation.
INSTRM.FOR	4.4.2	Main routine for doing instrumentation.
INSTRU.BAN	4.1	Menu for instrumenting source code.
INSTRU.COM	4	Com file for doing instrumentation.
INSTRU1.COM	4.2	Com file for selecting source file name.
INSTRU2.COM	4.3	Com file for defining instrumentation.
INSTRU3.COM	4.4	Com file for doing instrumentation.
INSTRU4.COM	4.5	Com file for editing instrumented source.
INTRO.COM	3	Com file for ACAP introduction.
INTRO1.COM	3.1	Com file for ACAP overview.
INTRO2.COM	3.2	Com file for ACAP flow.
INTRO3.COM	3.3	Com file for language availability intro.
IO.INC	-	Include file used in PRINRKG.
IRM.MTL	-	Master template file for instrumentation. Used by INSTRU2.
LENGTH.FOR	6.2.3.4.1.1 6.2.3.7.3.1	Function LENGTH finds the length of a string.

TABLE 4-1 (Continued)

ACAP PROGRAM FILES

Filename	Structure Chart Number	Description
MACUSE.FOR	5.4.9	Complies MACRO usage information for use by MACRPORT.FOR.
MACRPORT.FOR	6.2.3.9	Creates a MACRO usage report.
MACCOM.INC	-	Include file used by MACUSE.FOR and MACRPORT.FOR.
MAS.COM	5.4.7	Com file for creating master file.
MASTER.COM	8	Com file used in New Language.
MENU.NML	-	Namelist file containing $METRIC_MENU and $REP_WRITE_MENU used by REPEXEC and SUMEXEC.
MENUNML.INC	-	Include file for METRIC_MENU and REPWRITE_MENU.
METRICCOM.INC	-	Include file for METRICS namelist and common used by SQMETS.
METRICNML.INC	-	Include file for METRICS namelist.
MFCRE.BAN	5.4.7.1	Banner indicating creating master file.
MODREP.FOR	6.2.3.3	Main routine for generating Module Report.
NEXTPROC.COM	6.2.3	Com file for generating selected reports.
NMLEXEC.BAN	4.3.2	Banner indicating processing instrumentation namelist.
NOAVAIL.MEN	5.2.4	Banner indicating language not available.
NORMALIZE.FOR	5.4.2.3	Subroutine for normalizing a source line.
PAR11T.FOR	5.4.11.7	Parses a line for PDP11.
PAR68T.FOR	5.4.8.5	Subroutine for parsing M68000 source.
PARSER.FOR	5.4.2.5	Subroutine for parsing 6800 source.

TABLE 4-1 (Continued)

ACAP PROGRAM FILES

Filename	Structure Chart Number	Description
PARSKC.FOR	5.4.10.5	Subroutine for parsing SKC source.
P6800.TBL	-	Parser Table for 6800 source.
PM68000.TBL	-	Parser Table for M68000 source.
PDP11.MEN	5.2.6	Banner for PDP11 selection.
PARSERNML.INC	-	Include file for 6800 PARSER_TABLE.
PARSKCNML.INC	-	Include file for SKC PARSER_TABLE.
P6800TBL.MTL	-	Master template file for 6800 PARSER_TABLE.
PSKCTBL.MTL	-	Master template file for SKC PARSER_TABLE.
PAR68TNML.INC	-	Include file for PM68000 PARSER_TABLE.
PM68000TB.MTL	-	Master template file for M68000 PARSER_TABLE.
PATHPKG.FOR	6.2.3.6.3	Subroutine PATH and RECURSION.
PRINPKG.FOR	6.2.3.6.3.2	Subroutine PRINREP, STRUCTURE, WRITE_HEADER, CALCCOLS, BUILD_LINE, CHECK_RECURSION, LAST_CALL.
PRINTEXEC.FOR	7.2	Main routine for view/print.
PRINTEXEC.HLP	-	HELP file passed to GIVE_HELP in View/Print.
PSKC.TBL	-	SKC PARSER_TABLE.
REPEXEC.FOR	6.2.1	Main routine for selecting reports. Also contains SET_UP_RUN.
REPEXEC.HLP	-	Help file passed to GIVE_HELP by REP_EXEC.
REPORT.MTL	-	Master template file for Compliance and Summary Reports.

TABLE 4-1 (Continued)
ACAP PROGRAM FILES

Filename	Structure Chart Number	Description
RWEXEC.BAN	6.2.2	Banner indicating report writer executing.
SAEXEC.BAN	5.4.3	Banner indicating executing static analyzer.
SCAN11T.FOR	5.4.11	Translate PDP11 code.
SCAN68T.FOR	5.4.8	Main routine for M68000 code translator.
SCANNER.FOR	5.4.2	Main routine for 6800 code translator.
SCANSKC.FOR	5.4.10	Main routine for SKC code translator.
SCNSQMNML.INC	-	Include file for SCAN_SQM (file SCANSQM.NML) and ABSTRACT_SQM (file ABSTSQM.NML) namelists used to communicate between code translator and static analyzer.
SELECT.FOR	6.2.1.2.2.1.3	Subroutine SELECT_METRICS.
SELFILE.FOR	5.3.1	Main routine for selecting instrumented file.
SELREP.FOR	6.2.1.2	Subroutines SELECT_REPORTS and NEXT_MENU.
SELREPVP.FOR	7.2.2	Subroutine SELECT_REPORT_VP used in View/Print menu selection.
SELSRC.COM	5.3	Com file for selecting instrumented source file.
SEQLIST.FOR	6.2.3.4	Main routine for generating sequential listing.
SKC.MEN	5.2.5	Banner for SKC selection.
SORTCH.FOR	6.2.3.6.1.1	Subroutine SORTCH.
SQMETS.FOR	5.4.6	Main routine for compiling software quality metrics.
SQMREP.FOR	6.2.3.5	Main routine for generating ssoftware Quality Metrics Report.

TABLE 4-1 (Continued)

ACAP PROGRAM FILES

Filename	Structure Chart Number	Description
SSORT.FOR	5.4.5.1	Subroutine SSORT.
STRINGCOM.INC	-	Include file for SCANNER containing string info.
STRUC.INC	-	Include file used in RD_PROC, PRINREP, STRUCTURE, and CHECK_RECURSION.
STRUCTURE.FOR	6.2.3.6	Main routine for generating Structure Diagrams.
SUMCOM.INC	-	Include file for SUM_REP with report headings.
SUMEXEC.FOR	6.2.1.2.2.1	Subroutine SUM_EXEC to select summary metrics.
SUMEXEC.HLP	-	Help file passed to GIVE_HELP by SUM_EXEC.
SUMREP.FOR	6.2.3.7	Main routine for generating Summary and Compliance Reports.
TONML.FOR	4.3.3.1	Subroutine setting up instrumentation namelist.
TRANS.COM	8	Main routine for View/Print.
TRNSLCOM.INC	-	Include file for SCANNER containing variable that are written to translation file.
TRANSLTPL.FOR	4.3.3	Main routine for instrumentation namelist.
UNIQUE.FOR	6.2.3.6.1	Subroutine UNIQUE.
UNISORTS.FOR	6.2.3.2.1	Subroutine UNISORTS.
VARUSE.FOR	5.4.4	Main routine for compiling variable usage data.
VARUSENML.INC	-	Include file for VARUSE.NML part of master file.
VIEWPRINT.BAN	7.1	Banner for View/Print.
VIEWPR.COM	7	Com file for View/Print.

TABLE 4-1 (Continued)

ACAP PROGRAM FILES

Filename	Structure Chart Number	Description
VUREPORT.FOR	6.2.3.8	Main routine for generating Variable Usage Report.
VUSQMNML.INC	-	Include file for VU_SQM namelist used in the file VUSQM.NML to communicate between VARUSE and SQMETS.

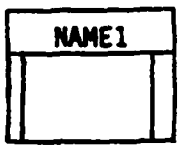
TABLE 4-2

LINK SEQUENCE FOR ACAP PROGRAMS

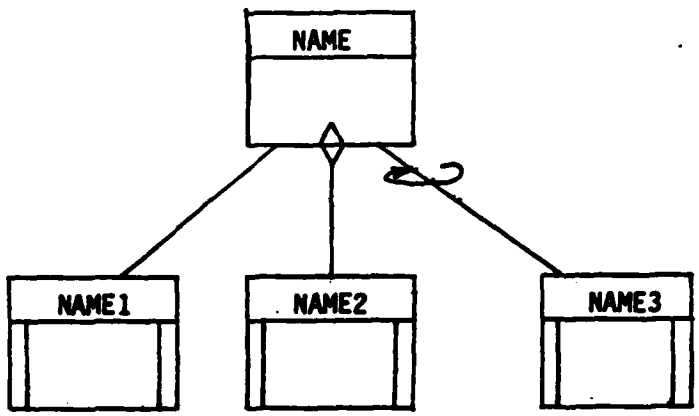
Program	Link Sequence
BRANCHES	BRANCHES, SSORT, BINSEARCH
CALLSRPT	CALLSRPT
EXTREP	EXTREP, UNISORTS, LENGTH
INSTRM	INSTRM, LENGTH, ERROR
MODREP	MODREP
MACUSE	MACUSE, SSORT
MACRPORT	MACRPORT
REPEXEC	REPEXEC, SELREP, CURSORPOS, SUMEXEC, DRAWMENU, SELECT, GIVEHELP, GETCHR, LENGTH, ERROR
PRINTEXEC	PRINTEXEC, SELREPVP CURSORPOS, DRAWMENU, GIVEHELP, GETCHR, LENGTH
SCANNER	SCANNER, NORMALIZE, BLK2BLK, PARSER, SSORT
SCAN68T	SCAN68T, NORMALIZE, BLK2BLK, PAR68T, SSORT
SCANSKC	SCANSKC, NORMALIZE, BLK2BLK, PARSKC, SSORT
SELFILE	SELFILE
SEQLIST	SEQLIST, LENGTH, NORMALIZE
SQMETS	SQMETS, BINSEARCH
SQMREP	SQMREP
SUMREP	SUMREP, LENGTH
TRANSLTPL	TRANSLTPL, TONML, LENGTH
VARUSE	VARUSE, SSORT
VUREPORT	VUREPORT
STRUCTURE	STRUCTURE, UNIQUE, SORTCH, CREPATH, PATHPKG, PRINPKG, LENGTH
SCAN11T	SCAN11T, PAR11T, NORMALIZE, SSORT

FIGURE 4-1
STRUCTURE CHART CONSTRUCTS

Independently Executable Module



Subroutine



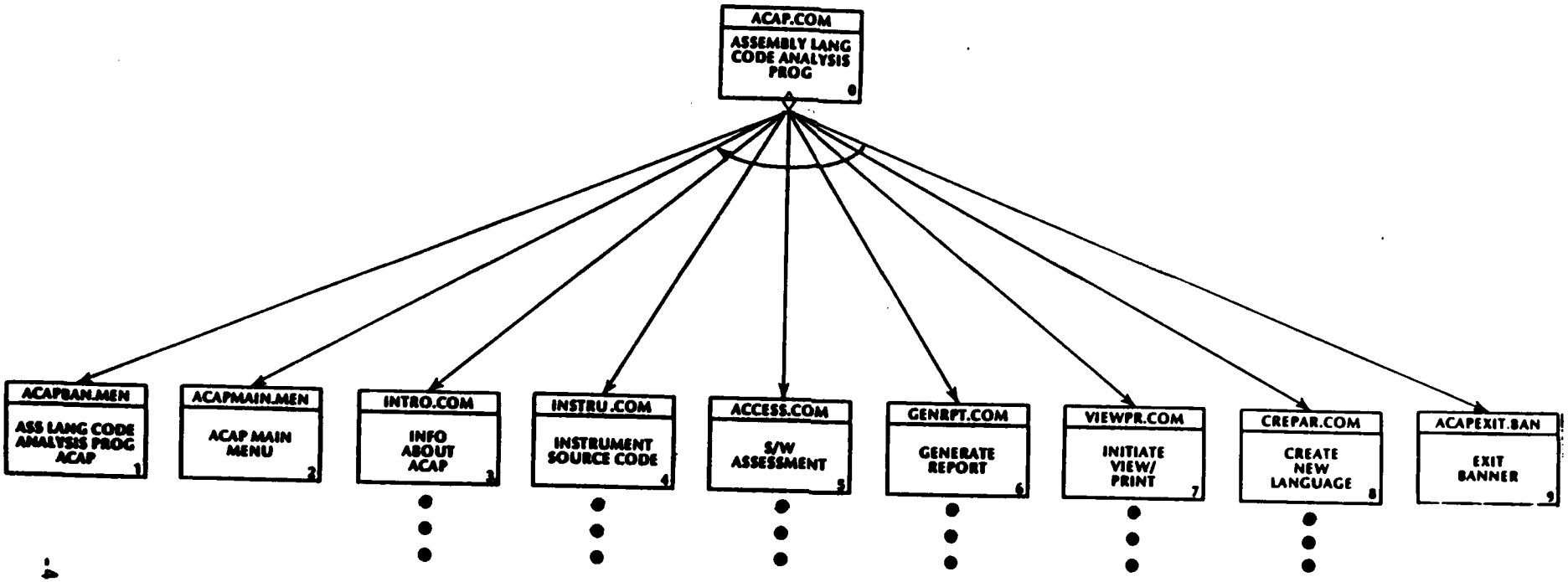
Subroutine NAME1 is always called by NAME.

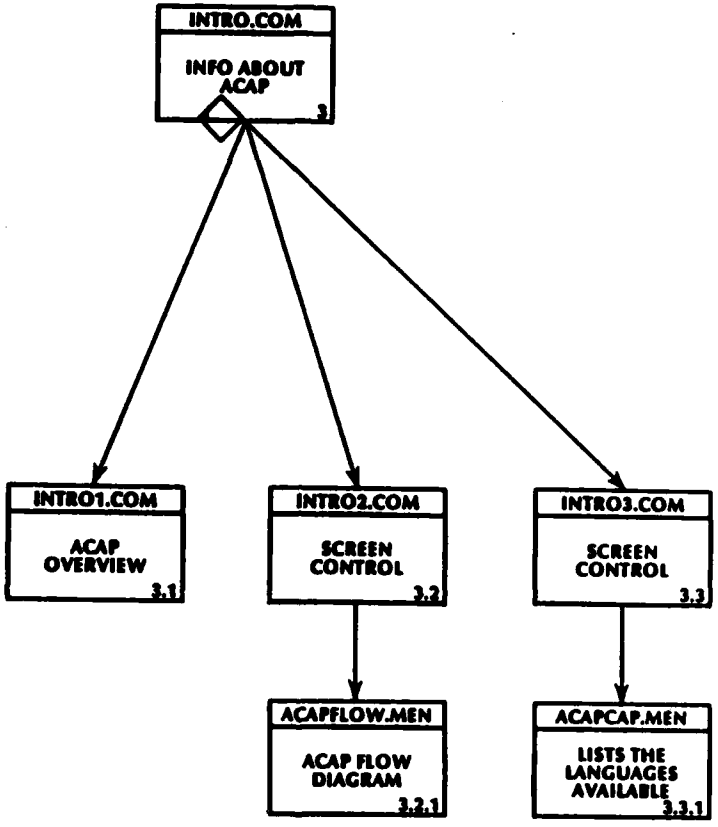
Subroutine NAME2 is conditionally called by NAME.

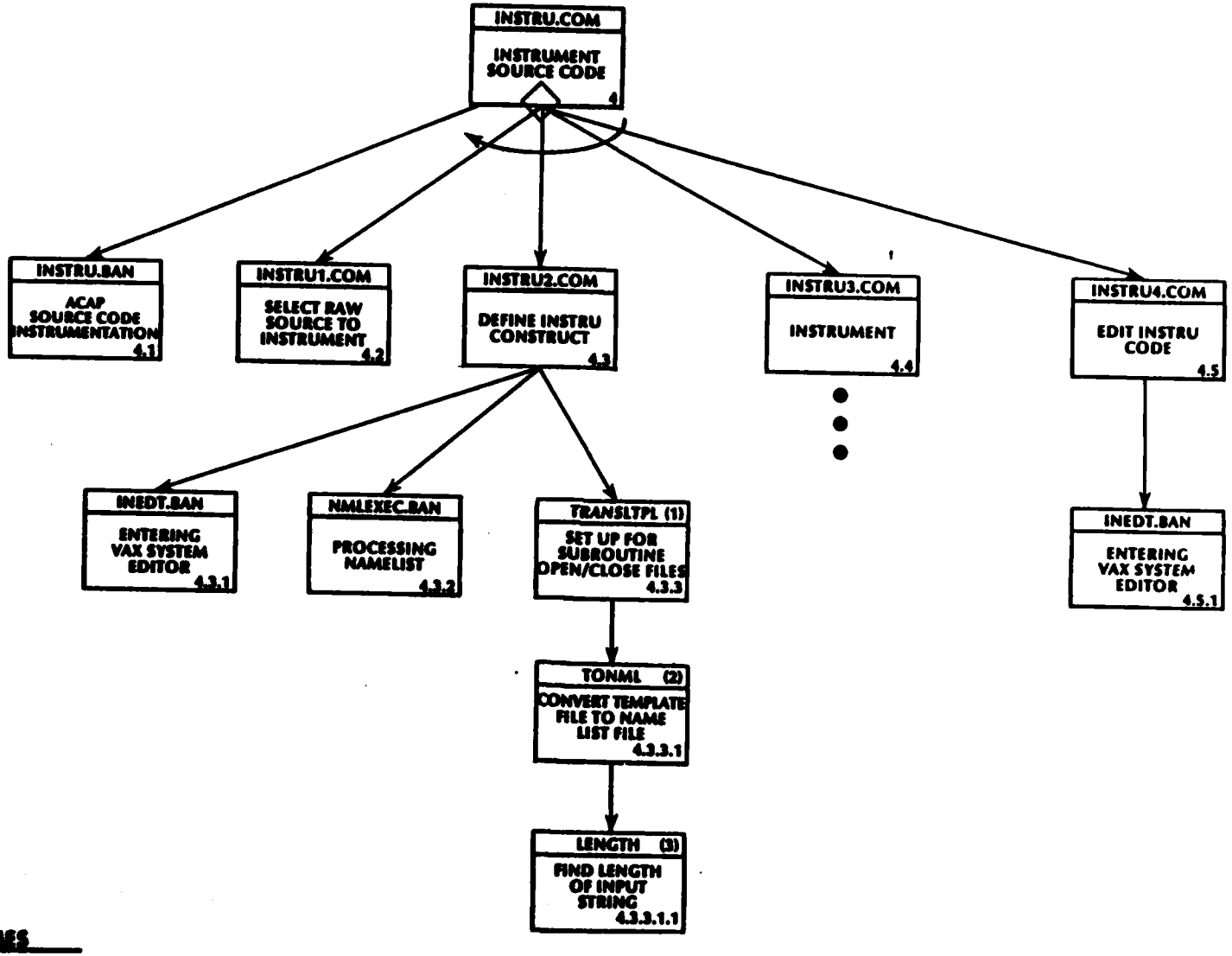
Subroutine NAME3 is always called a multiple number of time by NAME.

FIGURE 4-2
ACAP STRUCTURE CHARTS

NOTE: The name shown in the boxes is the actual Fortran name by which the procedure is referenced. If the procedure is part of a larger fortran file a note is made referencing the file name.

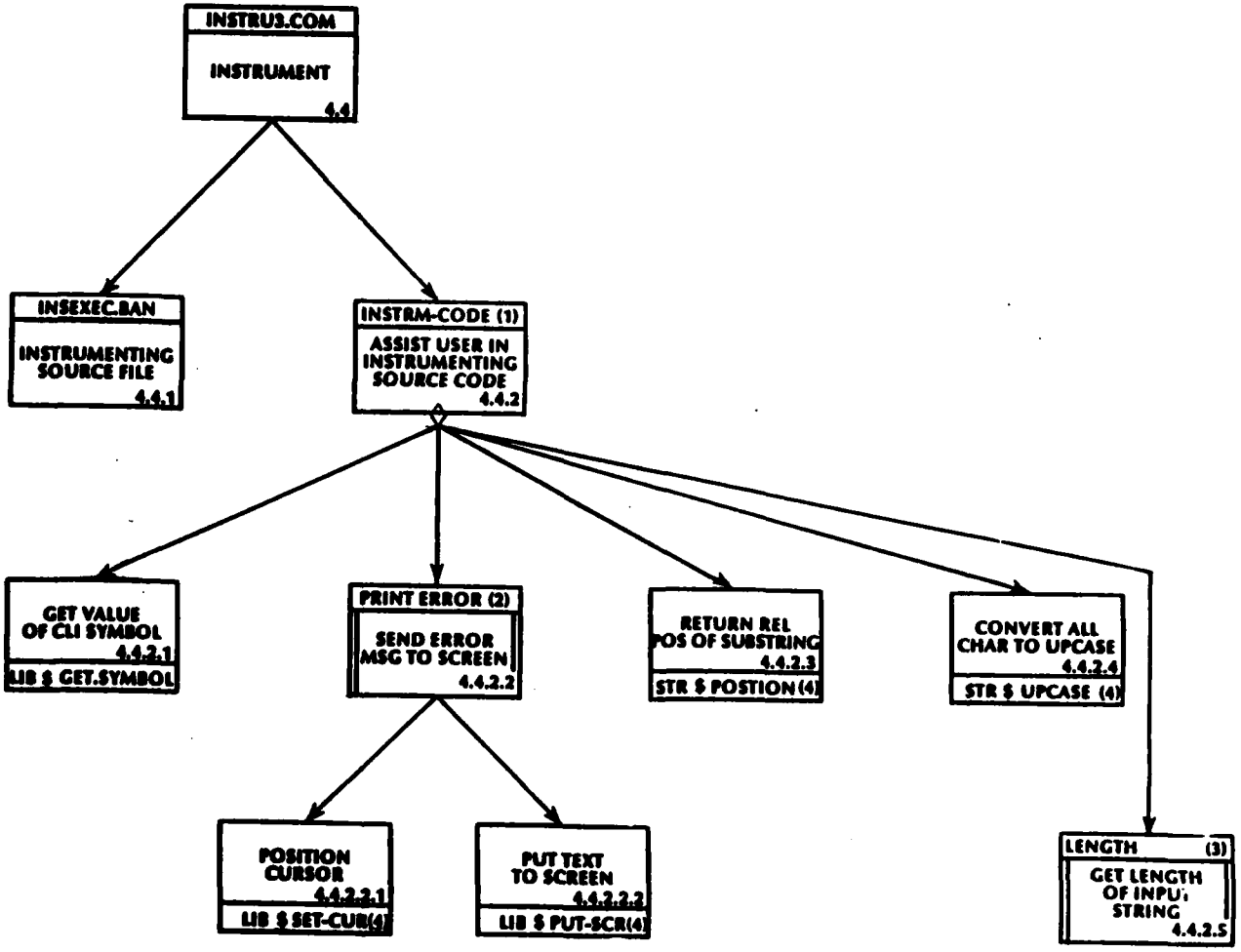

ACAP.COM
ASSEMBLY LANG CODE ANALYSIS PROG
0
ACAPBAN.MEN
ASS LANG CODE ANALYSIS PROG ACAP
1
ACAPMAIN.MEN
ACAP MAIN MENU
2
INTRO.COM
INFO ABOUT ACAP
3
INSTRU.COM
INSTRUMENT SOURCE CODE
4
ACCESS.COM
S/W ASSESSMENT
5
GENRPT.COM
GENERATE REPORT
6
VIEWPR.COM
INITIATE VIEW/ PRINT
7
CREPAR.COM
CREATE NEW LANGUAGE
8
ACAPEXIT.BAN
EXIT BANNER
9


INTRO.COM
INFO ABOUT ACAP
3
INTRO1.COM
ACAP OVERVIEW
3.1
INTRO2.COM
SCREEN CONTROL
3.2
INTRO3.COM
SCREEN CONTROL
3.3
ACAPFLOW.MEN
ACAP FLOW DIAGRAM
3.2.1
ACAPCAP.MEN
LISTS THE LANGUAGES AVAILABLE
3.3.1



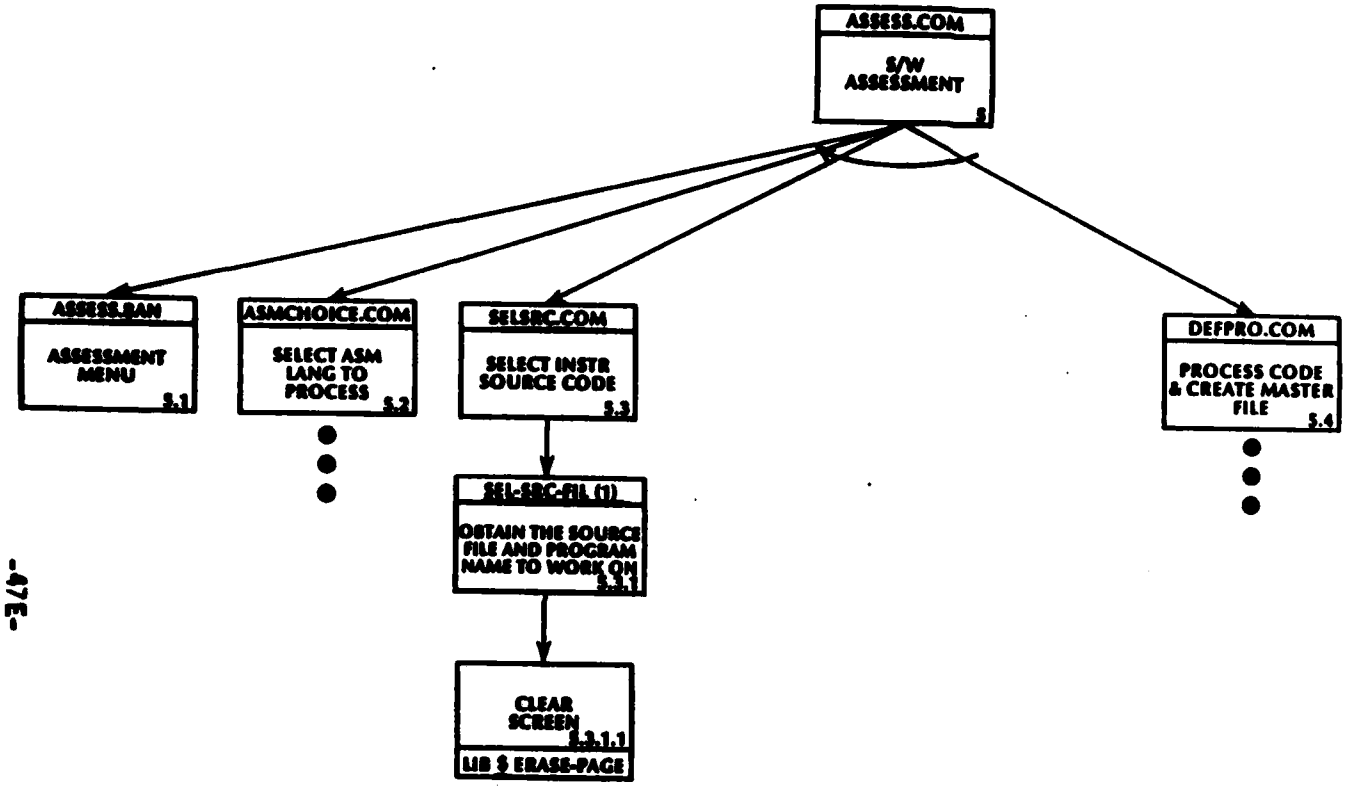
FILE NAMES

(1) TRANSLTPL.FOR
(2) TONML.FOR
(3) LENGTH.FOR



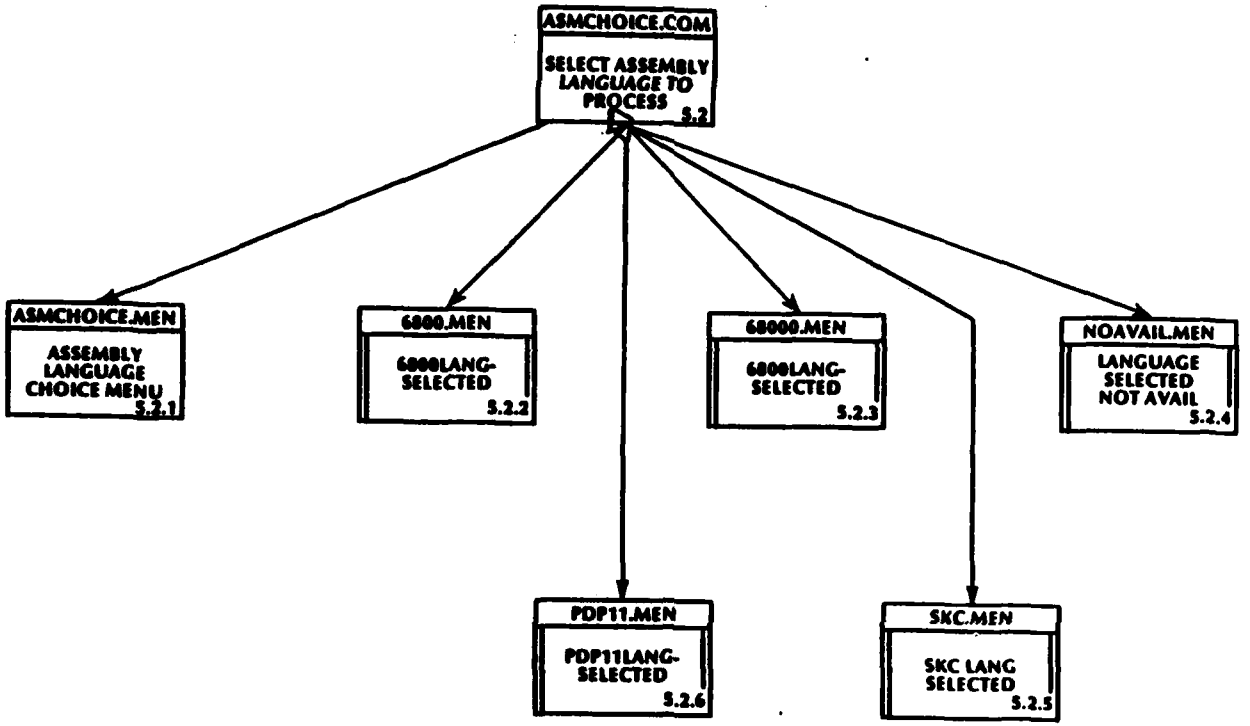
FILE NAMES

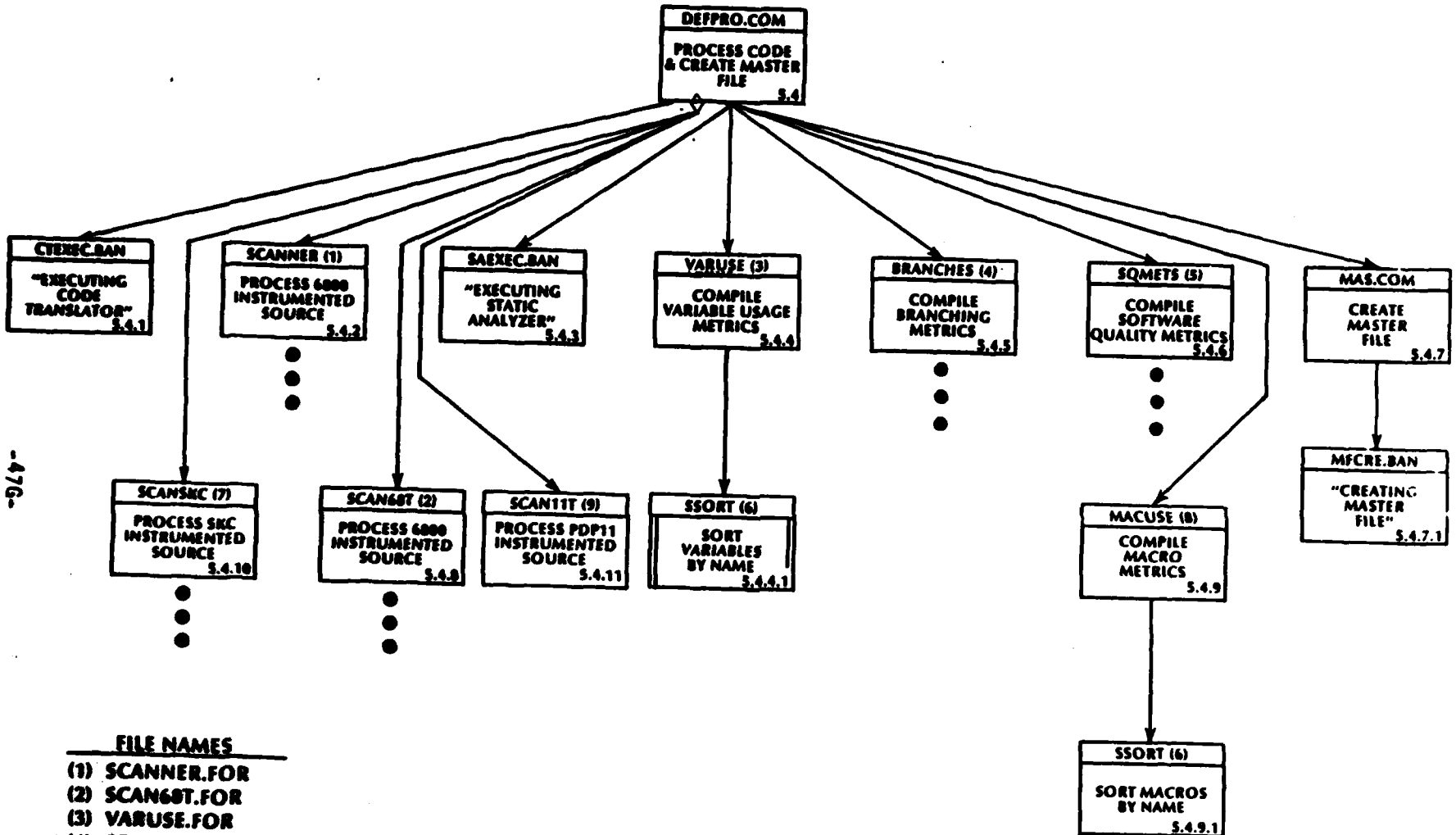
(1) INSTRM.FOR
(2) ERROR.FOR
(3) LENGTH.FOR
(4) SYSTEM ROUTINE



FILE NAMES

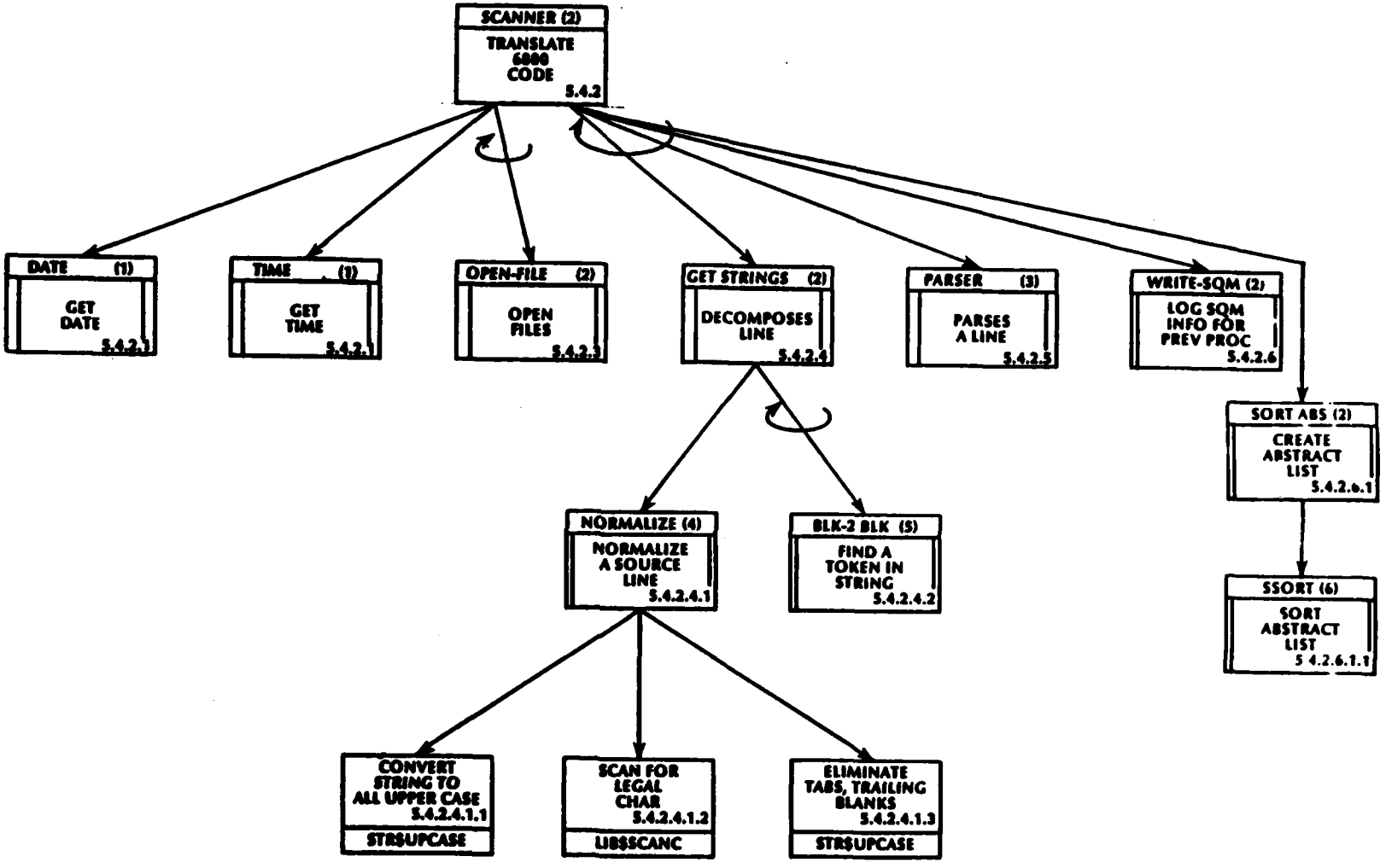
(1) SELFILE.FOR


ASMCHOICE.COM
SELECT ASSEMBLY LANGUAGE TO PROCESS
5.2
ASMCHOICE.MEN
ASSEMBLY LANGUAGE CHOICE MENU
5.2.1
6800.MEN
6800LANG-SELECTED
5.2.2
68000.MEN
6800LANG-SELECTED
5.2.3
NOAVAIL.MEN
LANGUAGE SELECTED NOT AVAIL
5.2.4
PDP11.MEN
PDP11LANG-SELECTED
5.2.6
SKC.MEN
SKC LANG SELECTED
5.2.5



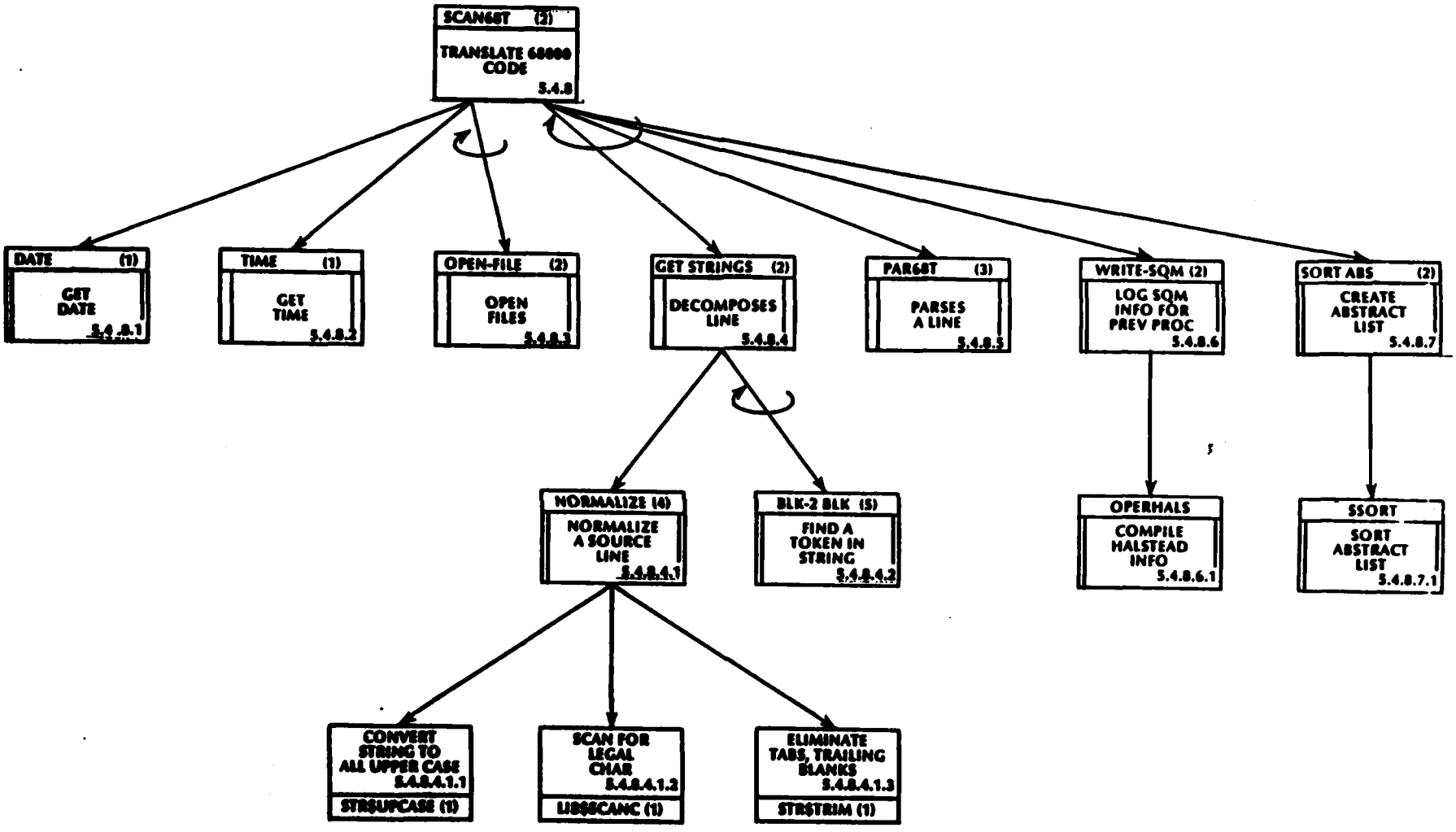
FILE NAMES

(1) SCANNER.FOR
(2) SCAN68T.FOR
(3) VARUSE.FOR
(4) BRANCHES.FOR
(5) SQMETS.FOR
(6) SSORT.FOR
(7) SCANSKC.FOR
(8) MACUSE.FOR
(9) SCAN11T.FOR



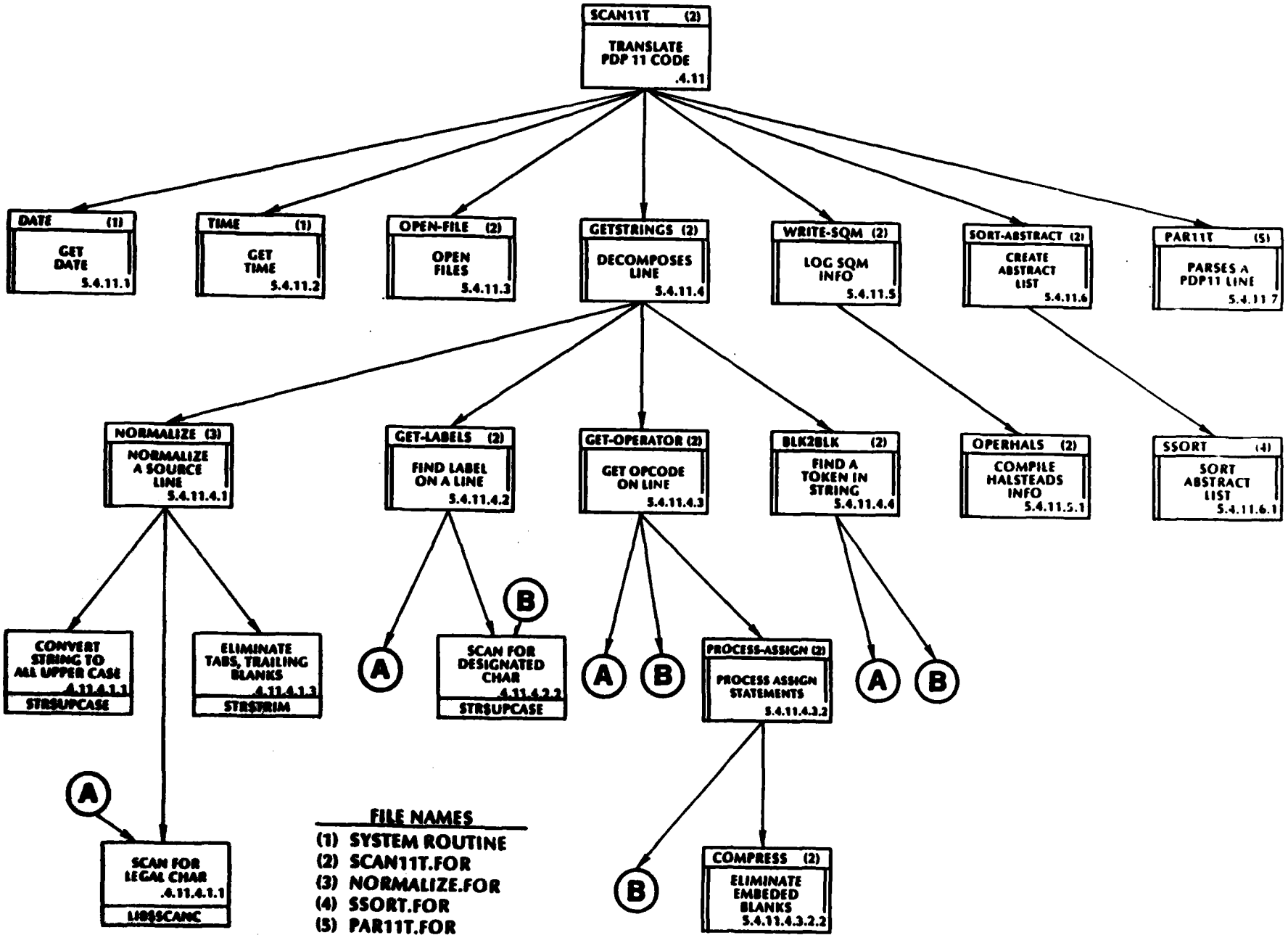
FILE NAMES

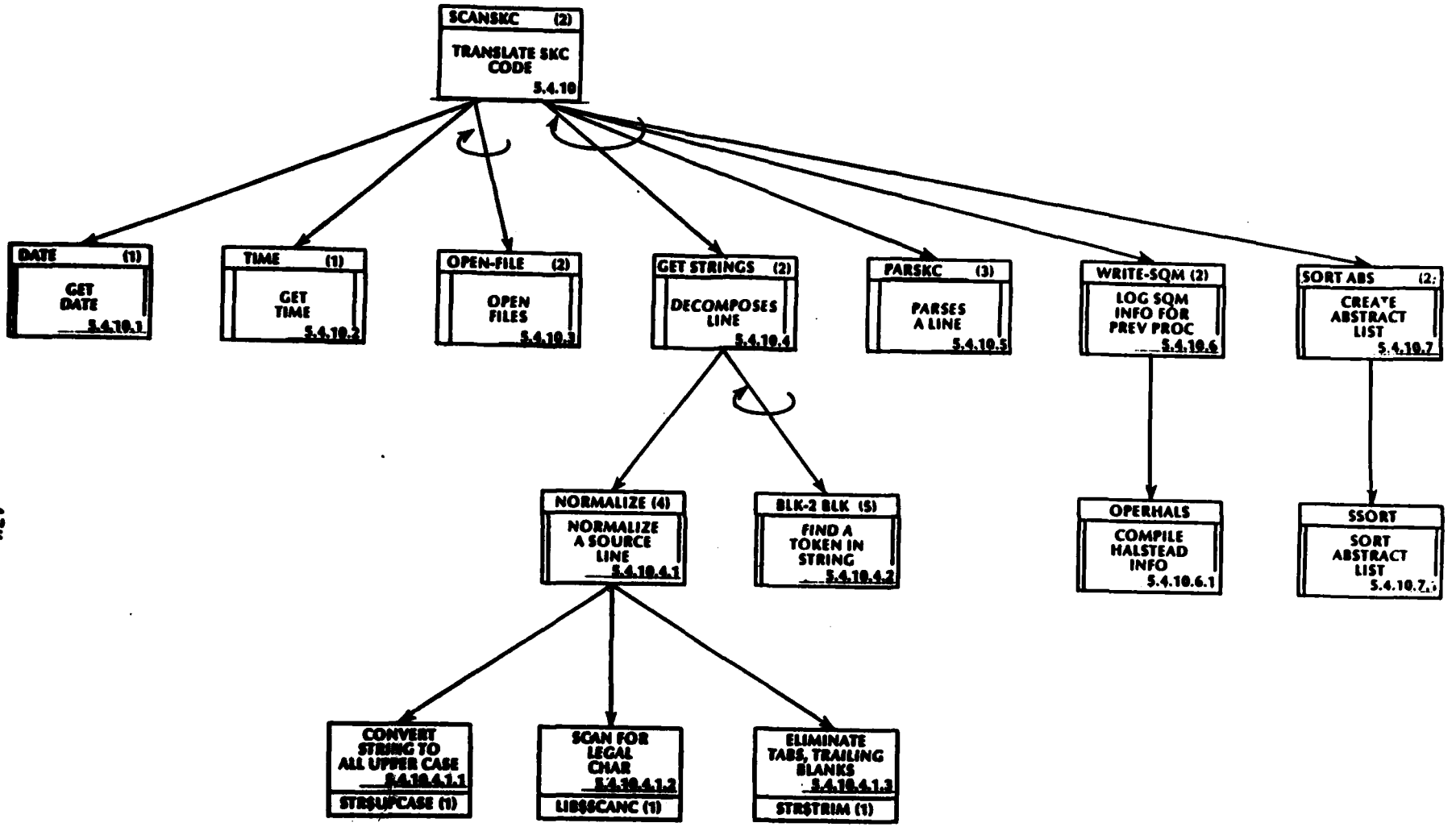
(1) SYSTEM ROUTINE
(2) SCAN68T.FOR
(3) PAR68T.FOR
(4) NORMALIZE.FOR
(5) BLK2BLK.FOR
(6) SSORT.FOR



FILE NAMES

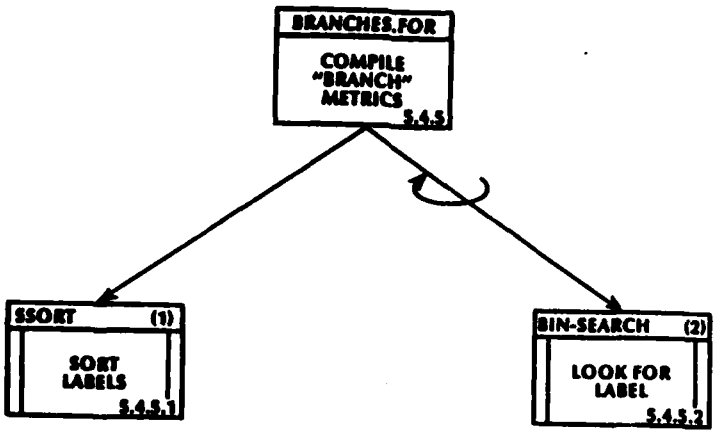
(1) SYSTEM ROUTINE
(2) SCAN68T.FOR
(3) PAR68T.FOR
(4) NORMALIZE.FOR
(5) BLK2BLK.FOR
(6) SSORT.FOR


SCAN11T (2)
TRANSLATE PDP 11 CODE .4.11
DATE (1)
GET DATE 5.4.11.1
TIME (1)
GET TIME 5.4.11.2
OPEN-FILE (2)
OPEN FILES 5.4.11.3
GETSTRINGS (2)
DECOMPOSES LINE 5.4.11.4
WRITE-SQM (2)
LOG SQM INFO 5.4.11.5
SORT-ABSTRACT (2)
CREATE ABSTRACT LIST 5.4.11.6
PAR11T (5)
PARSES A PDP11 LINE 5.4.11.7
NORMALIZE (3)
NORMALIZE A SOURCE LINE 5.4.11.4.1
GET-LABELS (2)
FIND LABEL ON A LINE 5.4.11.4.2
GET-OPERATOR (2)
GET OPCODE ON LINE 5.4.11.4.3
BLK2BLK (2)
FIND A TOKEN IN STRING 5.4.11.4.4
OPERHALS (2)
COMPILE HALSTEADS INFO 5.4.11.5.1
SSORT (4)
SORT ABSTRACT LIST 5.4.11.6.1
CONVERT STRING TO ALL UPPER CASE .4.11.4.1.1
STR$UPCASE
ELIMINATE TABS, TRAILING BLANKS .4.11.4.1.3
STR$TRIM
A
SCAN FOR LEGAL CHAR .4.11.4.1.1
LIB$SCANC
B
SCAN FOR DESIGNATED CHAR .4.11.4.2.2
STR$UPCASE
A
B
PROCESS-ASSIGN (2)
PROCESS ASSIGN STATEMENTS 5.4.11.4.3.2
A
B
B
COMPRESS (2)
ELIMINATE EMBEDED BLANKS 5.4.11.4.3.2.2
A
B
FILE NAMES
(1) SYSTEM ROUTINE
(2) SCAN11T.FOR
(3) NORMALIZE.FOR
(4) SSORT.FOR
(5) PAR11T.FOR

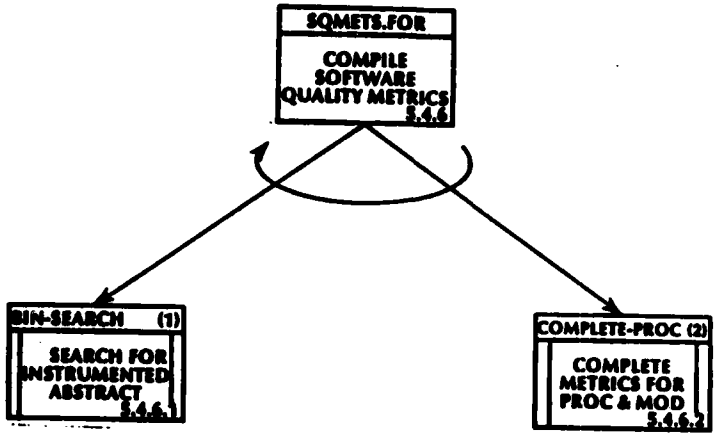


FILE NAMES

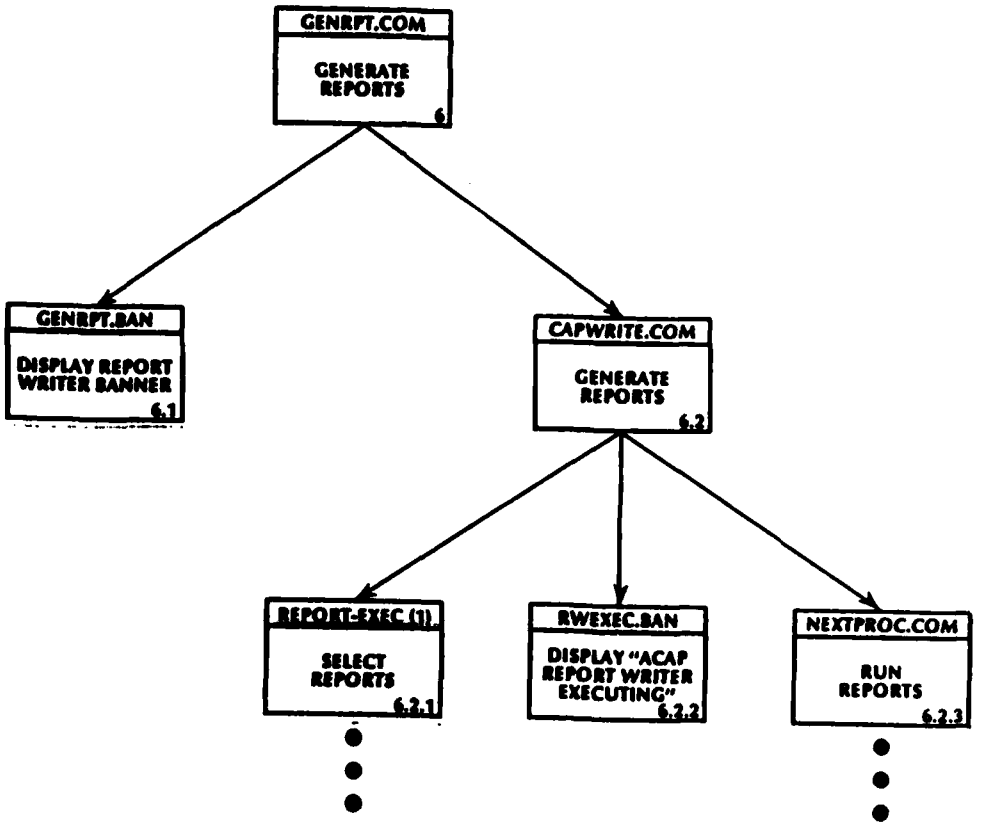
(1) SYSTEM ROUTINE
(2) SCANSKC.FOR
(3) PARSKC.FOR
(4) NORMALIZE.FOR
(5) BLK2BLK.FOR
(6) SSORT.FOR



FILE NAME	
(1)	SSORT.FOR
(2)	BINSEARCH.FOR

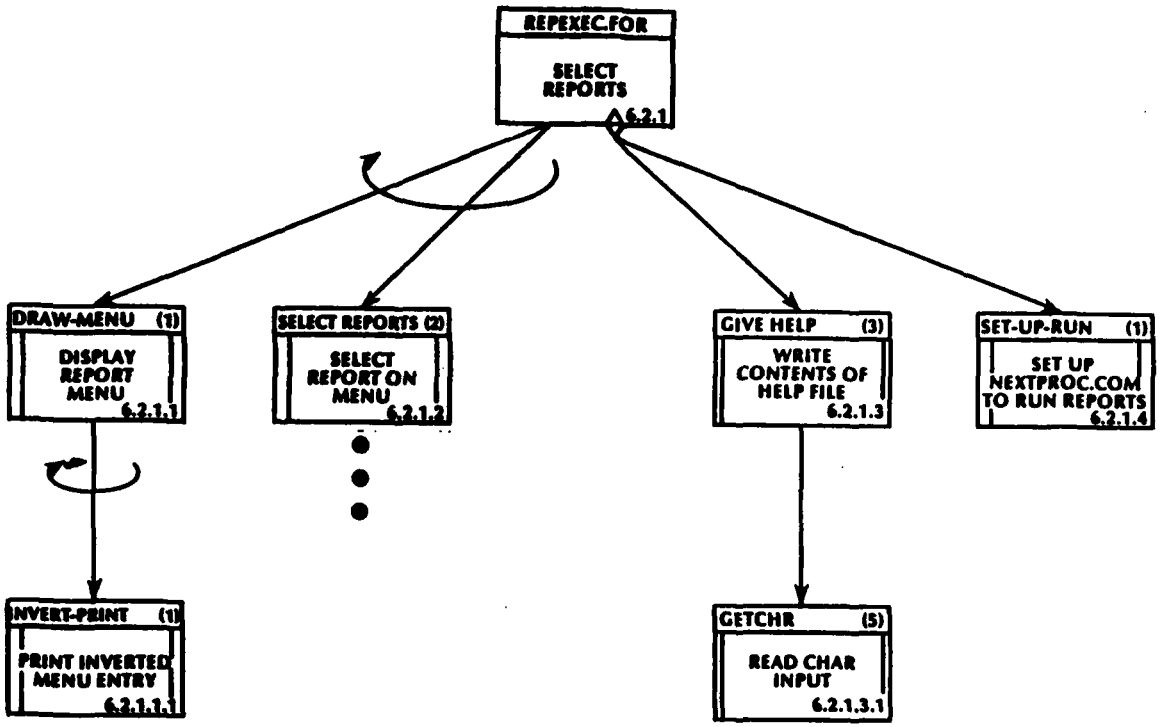


	FILE NAME
(1)	BINSEARCH.FOR
(2)	SQMETS.FOR



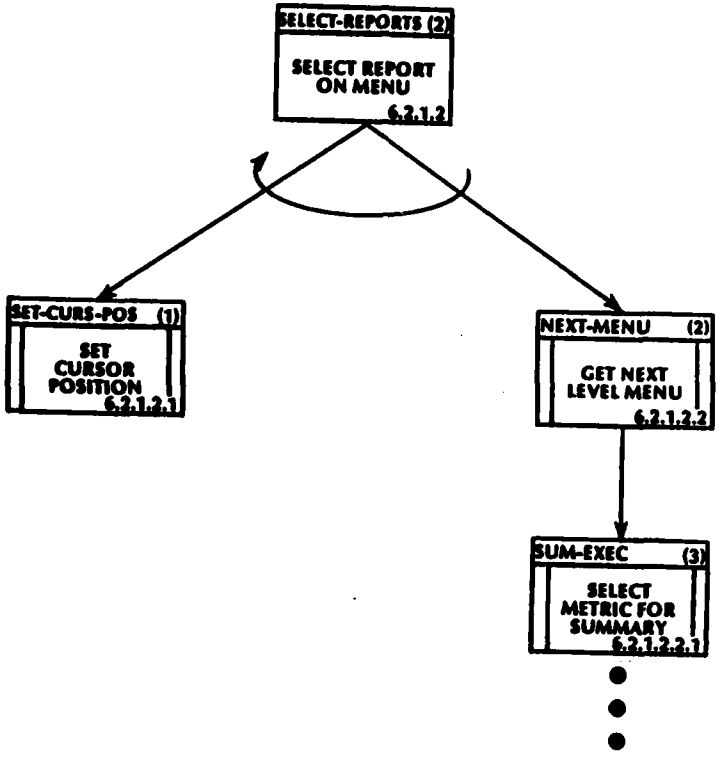
FILE NAMES

(1) REPEXEC.FOR



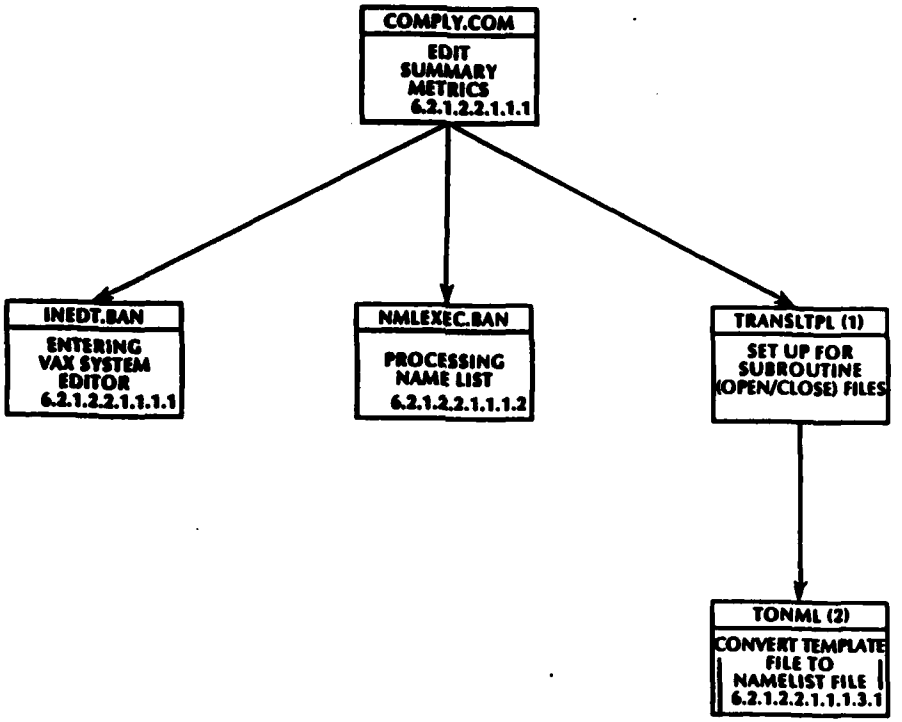
FILE NAMES

1) DRAWMENU.FOR
2) SELREP.FOR
3) GIVEHELP.FOR
4) REPEXEC.FOR
5) GETCHR.FOR



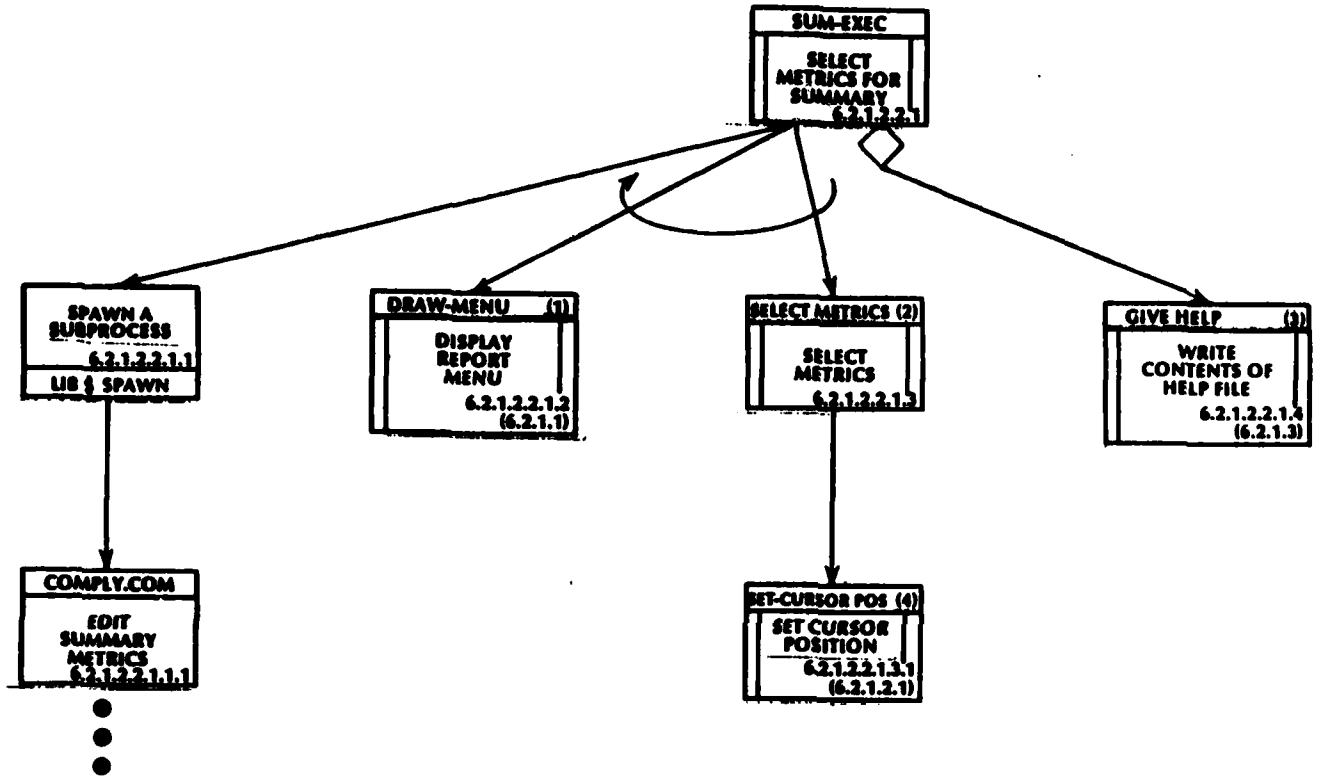
FILE NAMES

(1) CURSOR POS.FOR
(2) SELREP.FOR
(3) SUMEXEC.FOR



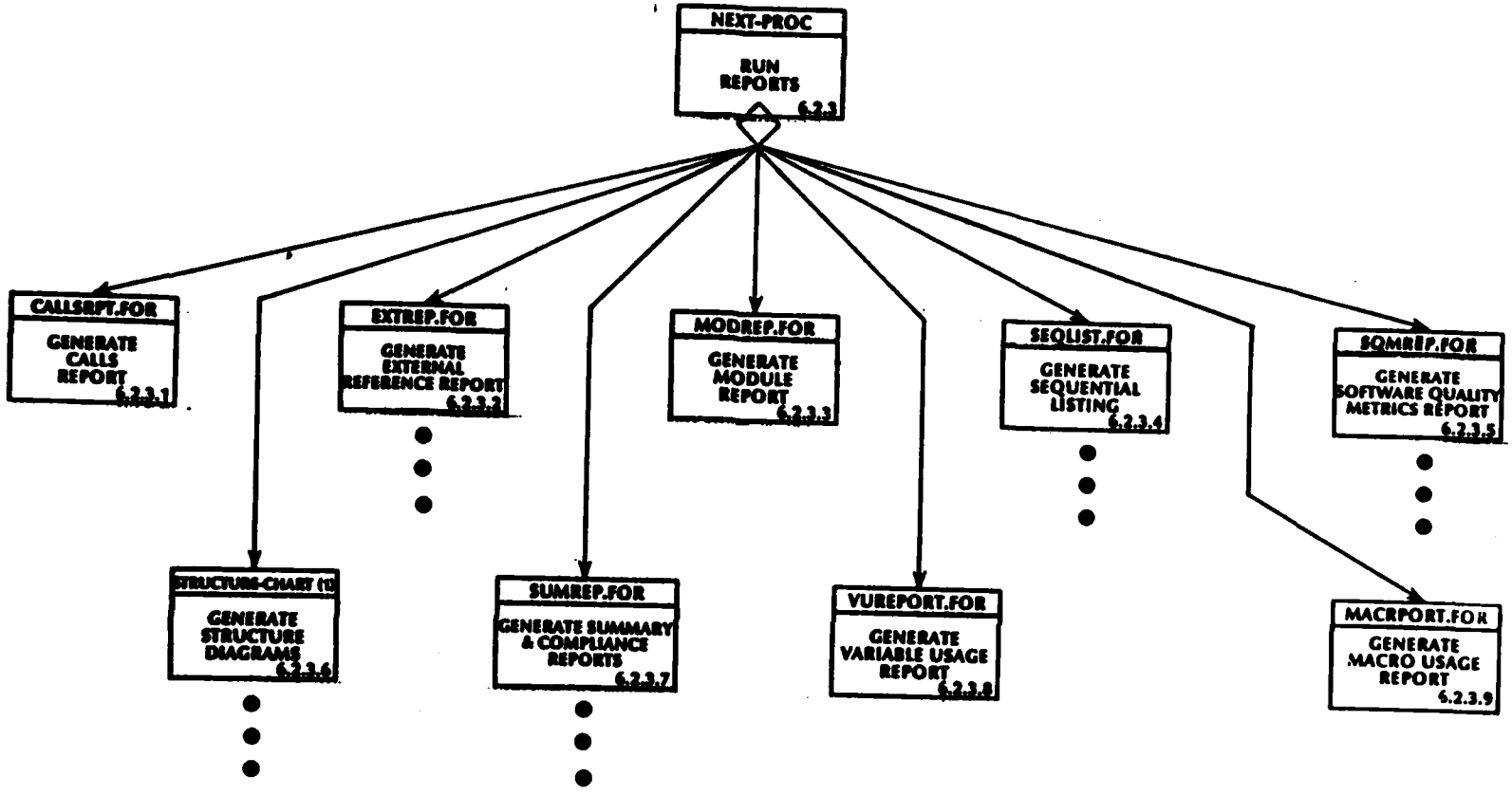
FILE NAMES

(1) TRANSLTPL.FOR
(2) TONML.FOR



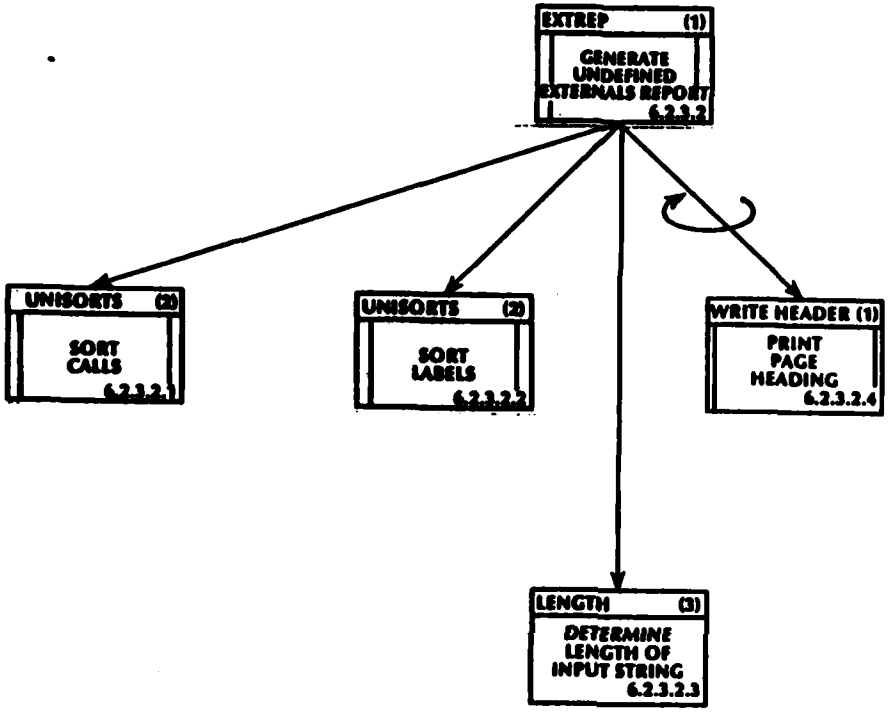
FILE NAMES

(1) DRAWMENU.FOR
(2) SELECT.FOR
(3) GIVEHELP.FOR
(4) CURSORPOS.FOR



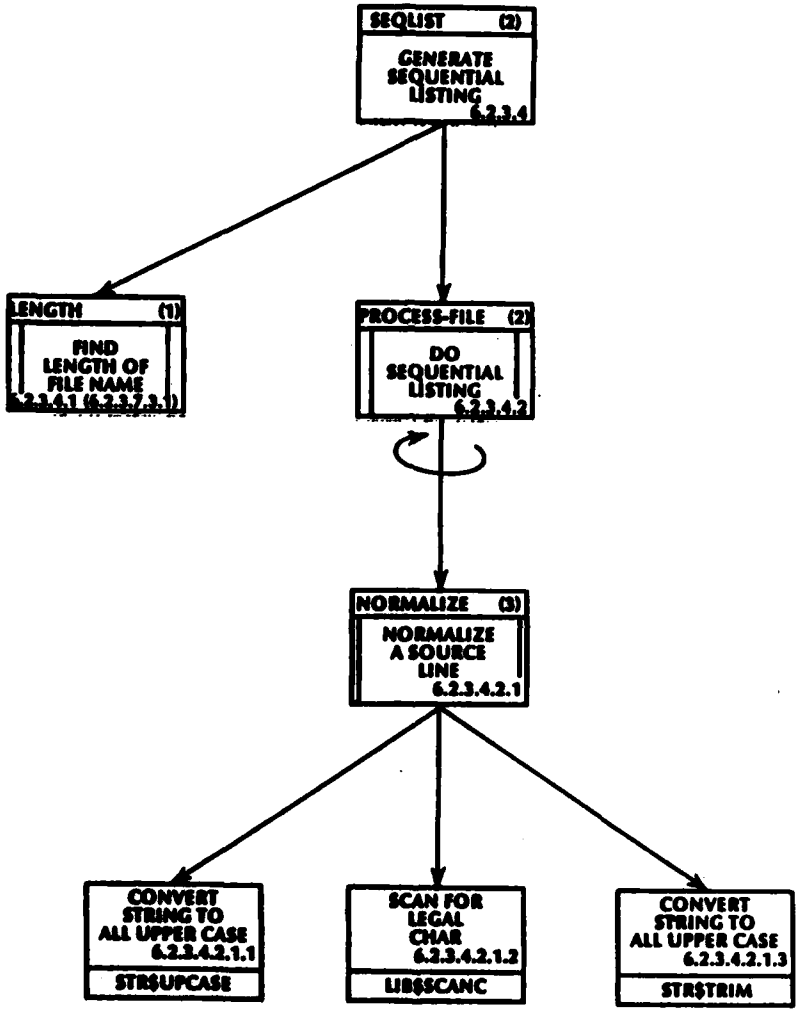
FILE NAME

(1) STRUCTURE.FOR



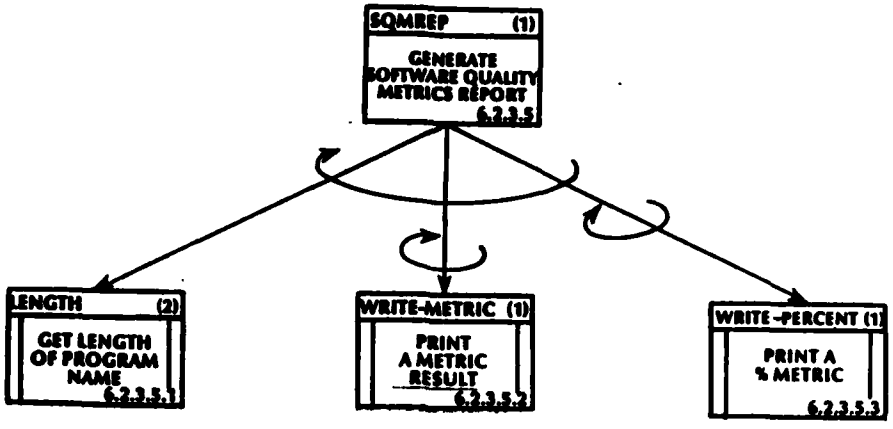
FILE NAMES

(1) EXTREP.FOR
(2) INISORTS.FOR
(3) LENGTH.FOR



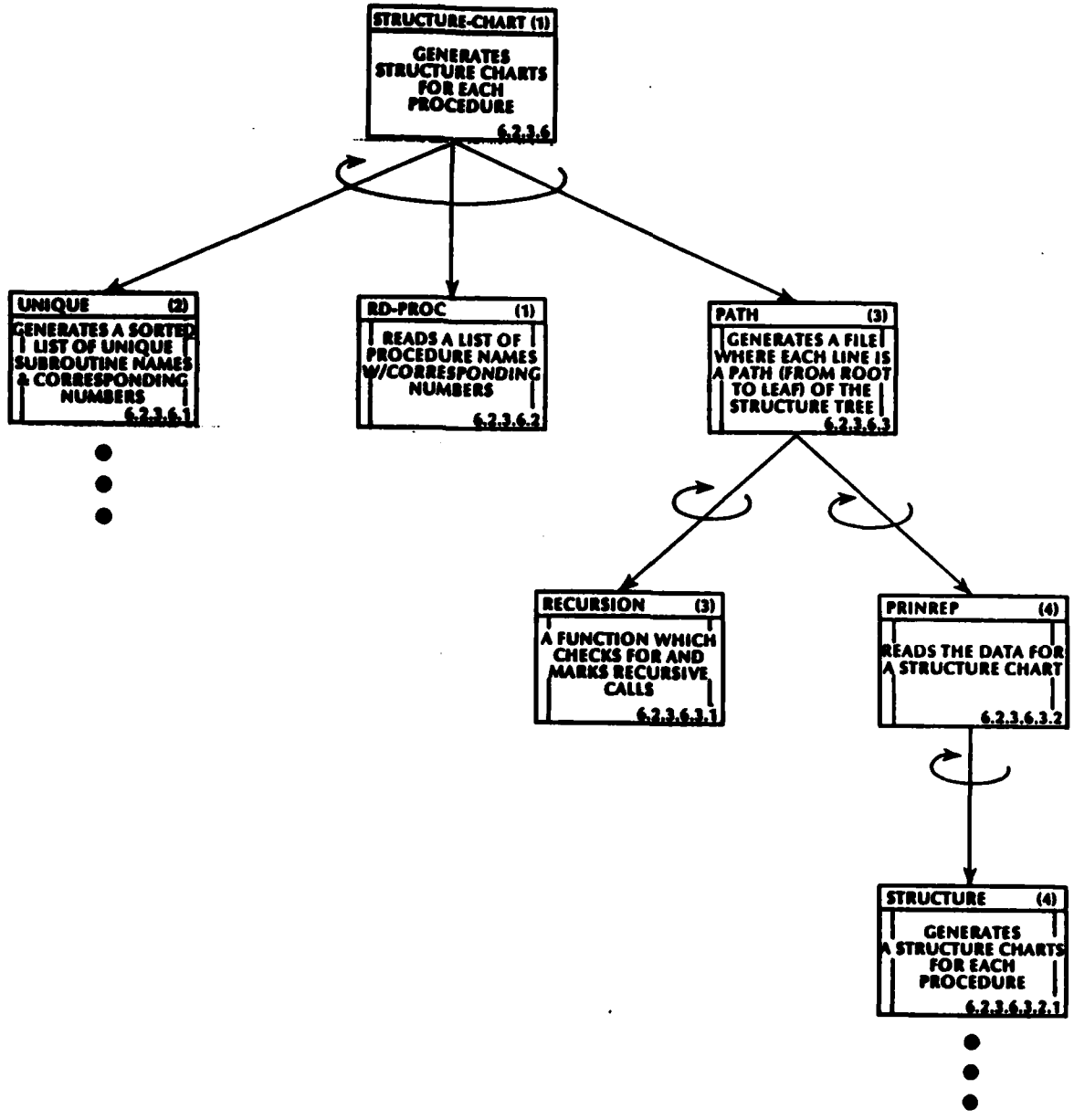
FILE NAMES

(1) LENGTH.FOR
(2) SEQLIST.FOR
(3) NORMALIZE.FOR

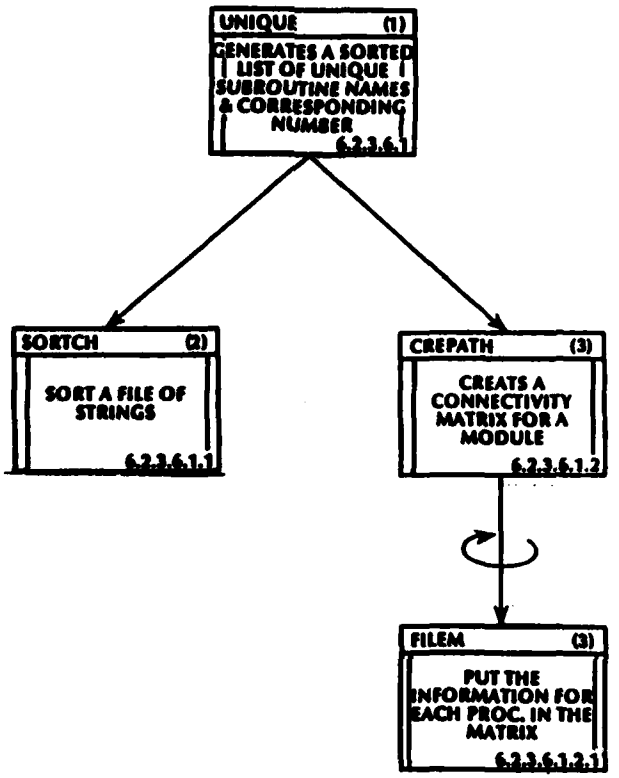


FILE NAMES

(1) SQMREP.FOR
(2) LENGTH.FOR

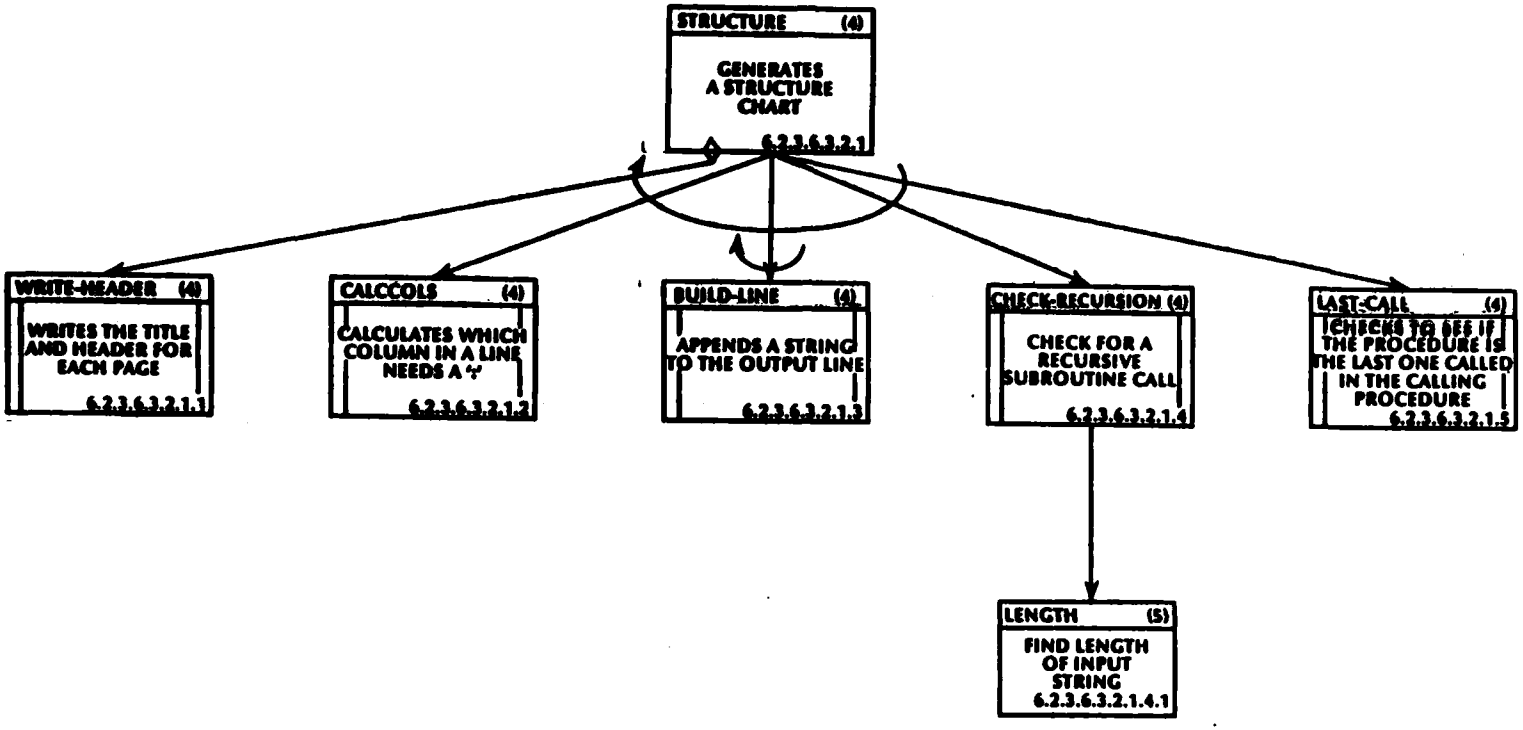


FILE NAMES
(1) STRUCTURE.FOR
(2) UNIQUE.FOR
(3) PATHPKG.FOR
(4) PRINPKG.FOR



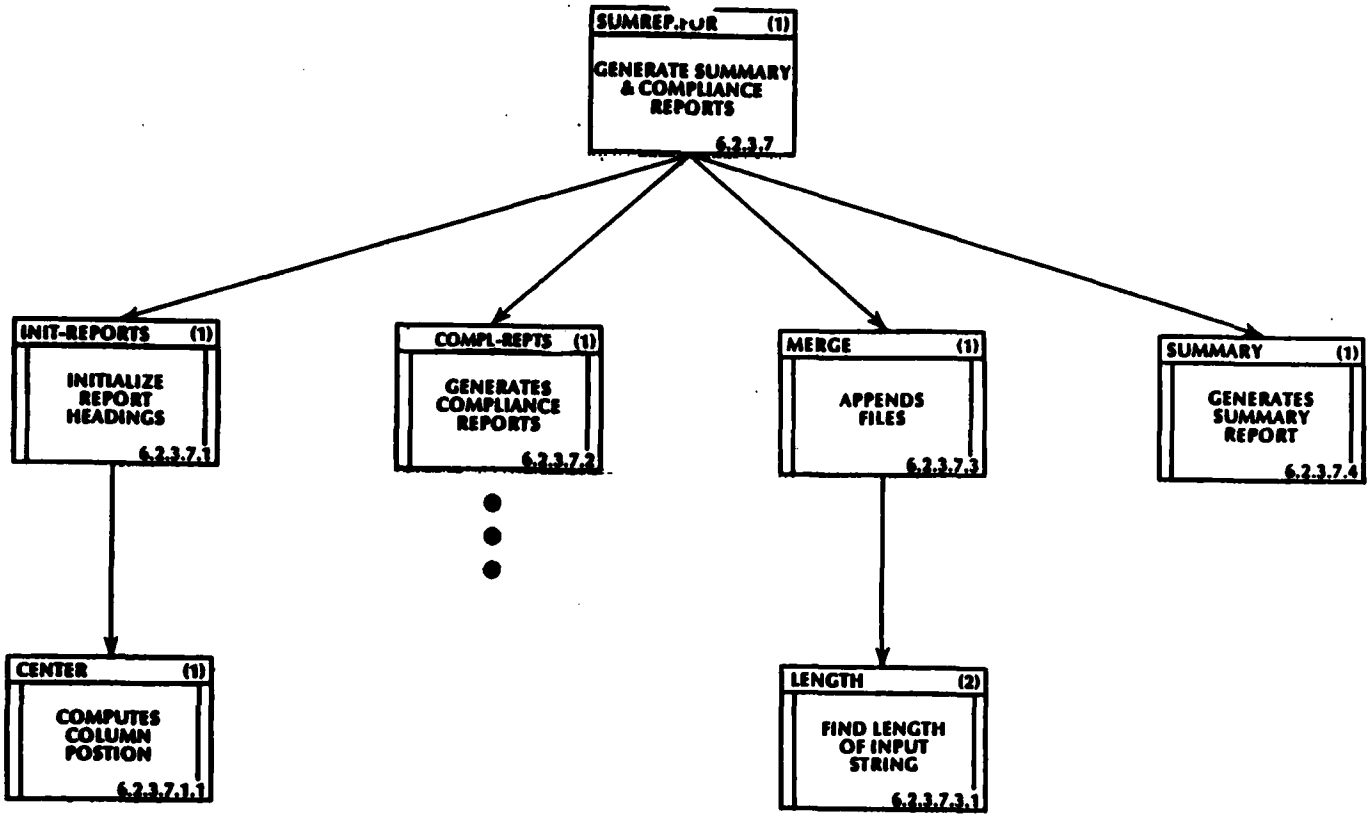
FILE NAMES

(1) UNIQUE.FOR
(2) SORTCH.FOR
(3) CREPATH.FOR



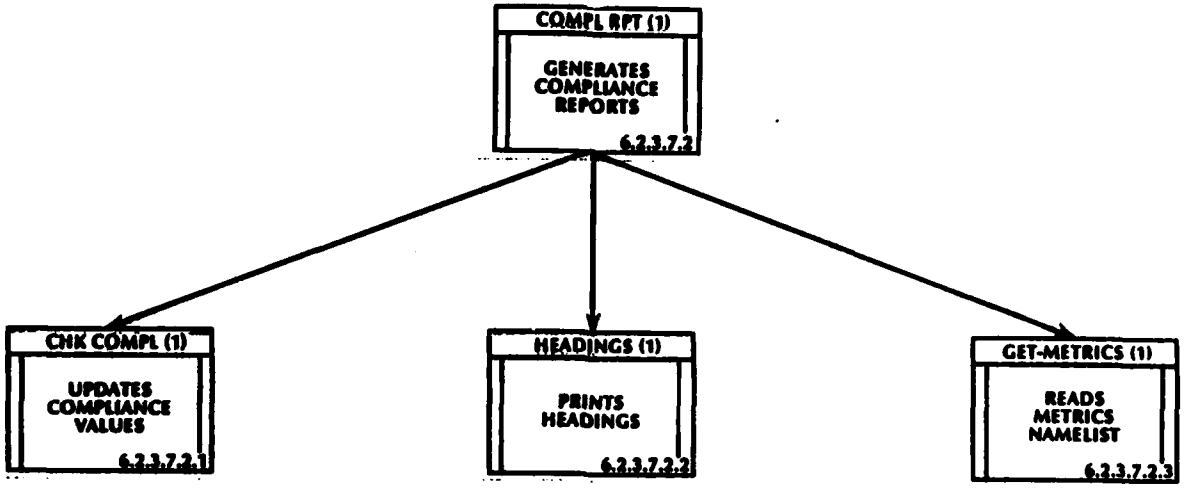
FILE NAMES

(4) PRINPKG.FOR
(5) LENGTH.FOR



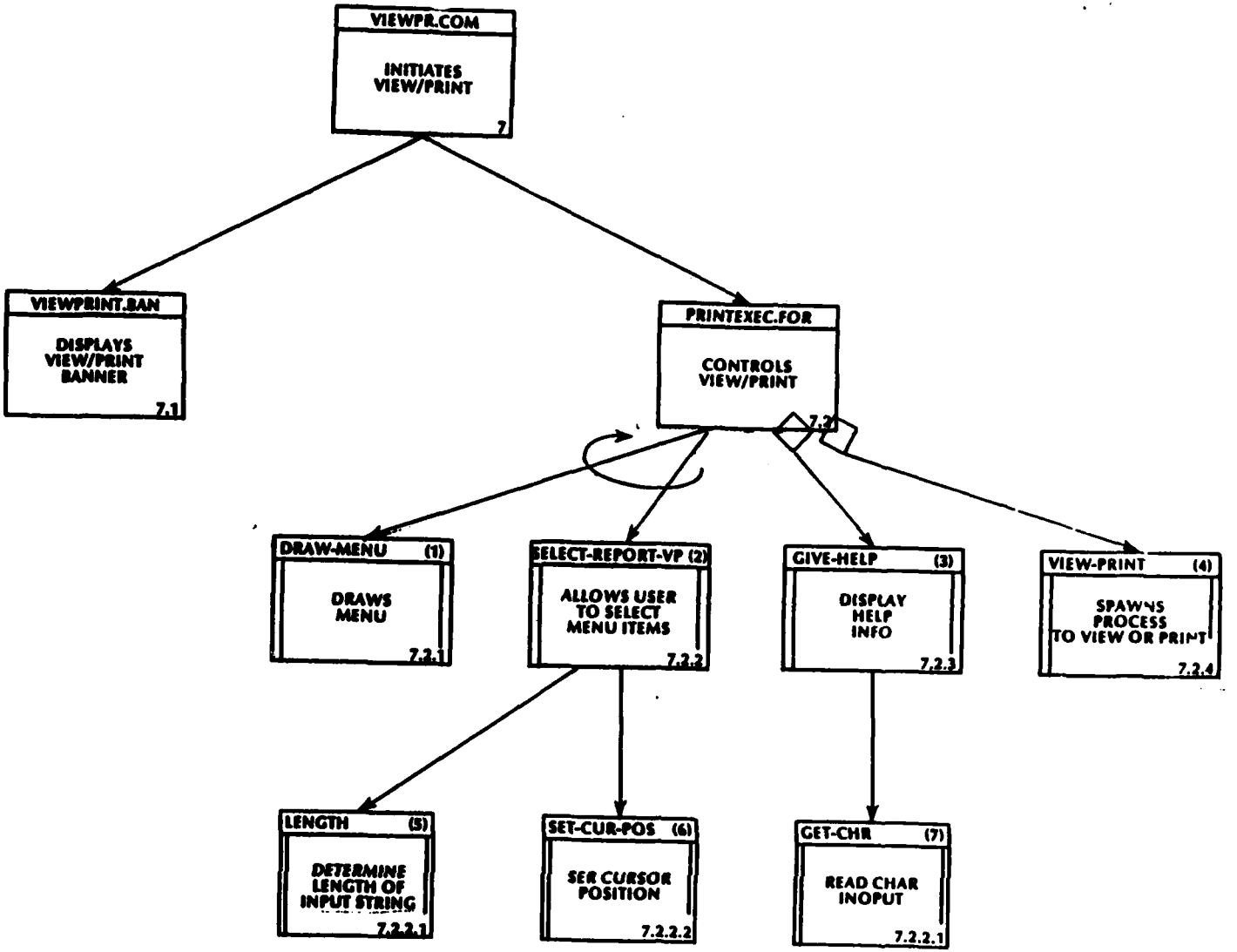
FILE NAMES

(1) SUMREP.FOR
(2) LENGTH.FOR



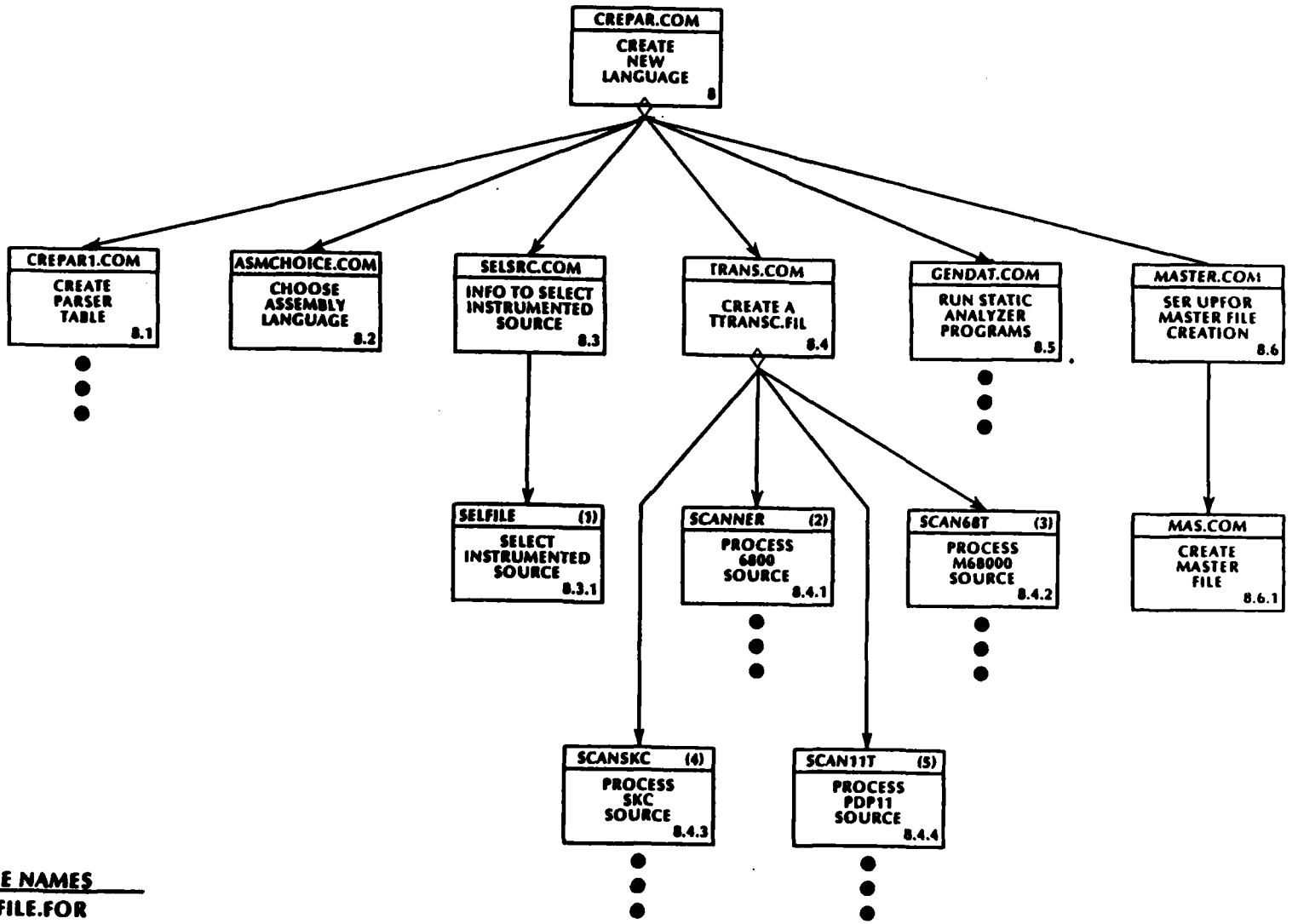
FILE NAMES

(1) SUMREP.FOR



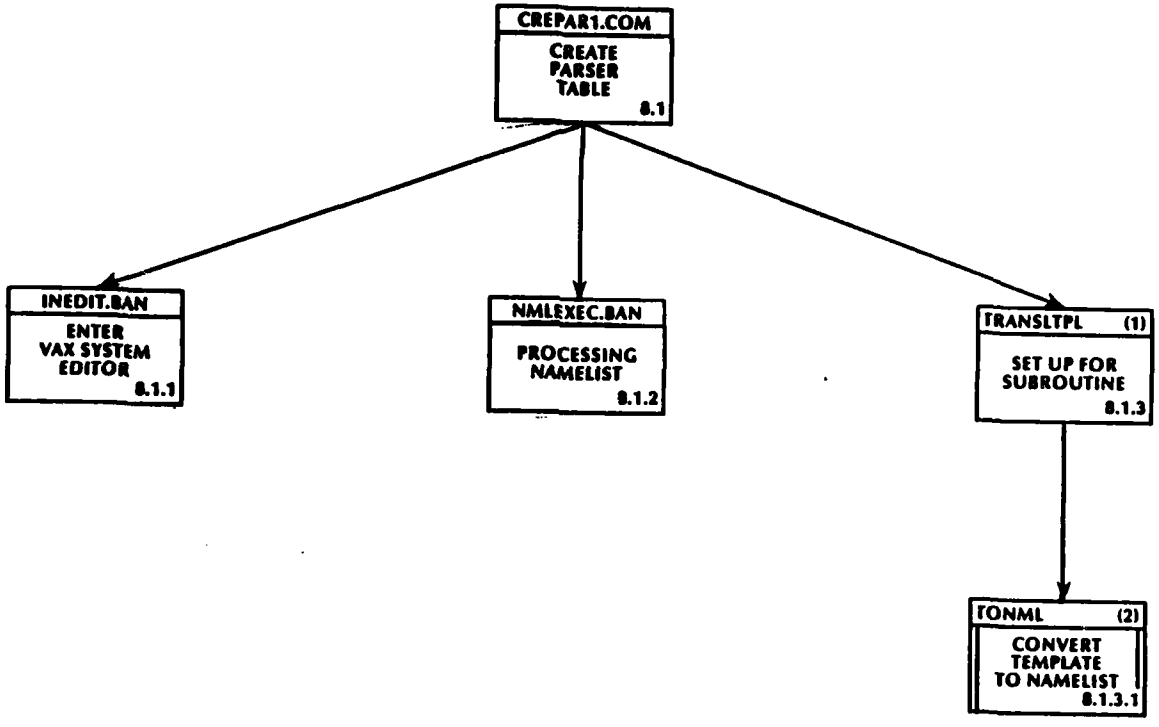
FILE NAMES

(1) DRAWMENUFOR
(2) SELPVP.FOR
(3) GIVEHELP.FOR
(4) PRINTEXEC.FOR
(5) LENGTH.FOR
(6) CURSORPOS.FOR
(7) GETCHR.FOR



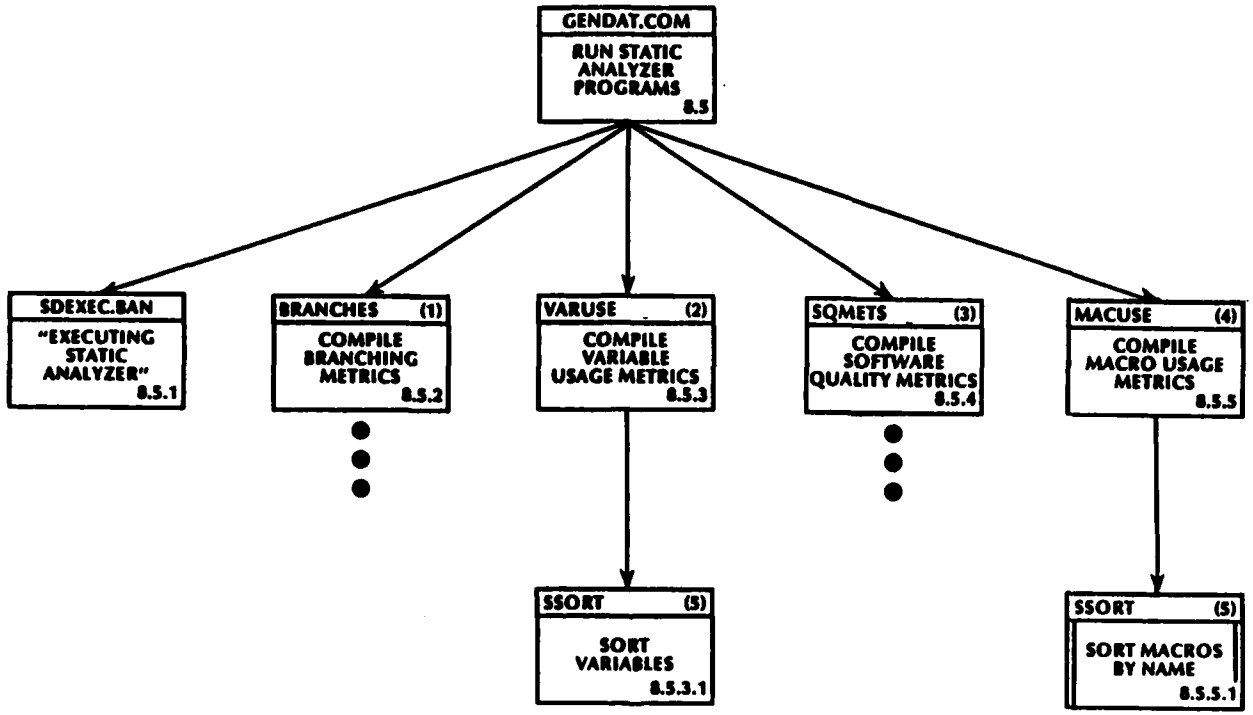
FILE NAMES

(1) SELFILE.FOR
(2) SCANER.FOR
(3) SCAN68T.FOR
(4) SCANSKC.FOR
(5) SCAN11T.FOR



FILE NAMES

(1) TRANSLTPL.FOR
(2) TONML.FOR



FILE NAMES

(1) BRANCHES.FOR
(2) VARUSE.FOR
(3) SQMETS.FOR
(4) MACUSE.FOR
(5) SSORT.FOR

5.0 ACAP DATA FILE DESCRIPTION

This section lists all the data files that are used by the ACAP system for communication between modules. Temporary files are not listed. The table gives the file name, where the file is used, where the file is created, and provides a brief description. Detailed formats for specific files associated with the Master File are presented in Appendix A.

(Note: SCANNER is used here in the generic sense and stands for either SCANNER.FOR, SCAN68T.FOR or SCANSKC.FOR.)

ACAP Files With Fixed File Names

File	Used	Created	Description
ABSTSQM.NML	SQMETS	SCANNER	Information on instrumented abstracts - number of lines and number of lines skipped.
BRANCH.NML	MAS	BRANCHES	Branch data for each procedure. Put into master file.
BRCHSQM.NML	SQMETS	BRANCHES	Information on branches - number of forward, backward, out, conditional, and unconditional branches; number of nodes and paths.
COMMENT.FIL		SCANNER	Optional debug data on comments controlled by COMMENT_PRNT in PARSER_TABLE.
CONTROL.NML	MAS	SCANNER	Data used to control analysis. Includes PARSER_TABLE. Put into master file.
ERROR.NML	MAS	SCANNER SQMETS BRANCHES VARUSE	Errors accumulated during analysis run. Put into master file.
INSTRM.FIL		SCANNER	Optional debug data on instrumented lines controlled by INSTRM_PRNT in PARSER_TABLE.

ACAP Files With Fixed File Names (Continued)

File	Used	Created	Description
IRM.NML	INSTRM	INSTRU2	Instrumentation file containing the instrumentation value and code constructs where value is to be inserted. Created from IRM.MTL if IRM.TPL does not exist.
MACROS.NML	MACRPORT	MACUSE	MACRO definition and usage information.
PARSER.TBL	SCANNER		Parser table containing a description of assembly language being analyzed.
PARSER.FIL		SCANNER	Optional debug data on parser performance controlled by PARSER_PRNT in PARSER_TABLE.
SCANSQM.NML	SQMETS BRANCHES	SCANNER	Software metrics accumulated by SCANNER for each procedure - dealing with instrumentation such as module name and sequence, procedure name and sequence, number of lines skipped, number of unique operators, number of operators used, number of unique operands, number of operands used, and the line number in the instrumented source on which the procedure starts.
SQM.NML	MAS	SQMETS	Data on software quality metrics. Put into master file.
STRINGS.FIL		SCANNER	Optional debug data on strings found in each line. Controlled by STRINGS PRNT in PARSER_TABLE.
TRNSL.FIL	SQMETS VARUSE BRANCHES	SCANNER	Translated assembly language code in generic form.
VUR.NML	MAS	VARUSE	Data on variable usage. Put into master file.

ACAP Files User-Defined File Names

File	Used	Created	Description
VUSQM.NML	SQMETS	VARUSE	Data on number of variables in a procedure.
Instrumented Source File	SCANNER SEQLIST	INSTRM	Assembly language source file instrumented so it is ready for analysis.
Master File	CALLSRPT EXTREP MACRPORT MODREP PRINTEXEC REPEXEC SEQLIST SQMREP STRUCTURE SUMREP VUREPORT	MAS	Concatenation of VUR.NML, BRANCHES.NML, SQM.NML, MACRO.NML, CONTROL.NML, ERROR.NML. (See Appendix A.)
Program_Name.CAL		CALLSRPT	Calls Report
.CPL		SUMREP	Compliance and Summary Reports
.EXT		EXTREP	Undefined Externals Reports
.MOD		MODREP	Module Report
.SQM		SQMREP	Software Quality Metrics Report
.STR		STRUCTURE	Structure Diagrams
.VUR		VUREP	Variable Usage Report
.MUR		MACRPORT	MACRO Usage Report
Source File	INSTRM		

6.0 SAMPLE REPORTS

The various reports available from ACAP are displayed in this section. These reports are obtained by using the generalized report writer described in Section 2.4.

```
                    C A P SEQUENCED SOFTWARE LISTING
C** ST_PROCEDURE MOD1_SIXTEENCHAR INST_PROC_SIXTEN
 ********************************************
 *Unit InSTRUCTIONS(INSTLST) - Header
 *
 * This is a test of all 68000 instructions
 *
 ********************************************
 * FIRST DO SOME INITIALIZATION
 ********************************************
         TTL
         OPT      A,BRL,CL,CRE
         SPC
         LIST
         NOLIST
         PAGE
         SECTION 1
         ORG      2000
FORNUM   SET      1000
         FAIL     !FOR KICKS
         XDEF     ADEF1,ADEF2,ADEF3
         XREF     DEF1,DEF2,DEF3
         INCLUDE PROG1
         INCLUDE PROG2
MAXTIM   EQU      @2000
CPU:     DC.B     0
         DC.B    -1
GSSIO:   DC.W     0
         DC.W    -1
TIMETAB  DS.B     MAXTIM
TIMTSK   DS.L     MAXTIM

 ********************************************
INST_PROC_SIXTEN EQU     *
         ABCD     D0,D1
         ABCD     -(A0),-(A1)
         ADD      LABEL1,D2
         ADD      D2,LABEL1
         ADDA     LABEL2,A2
         ADDI     #IMMED1,LABEL3
         ADDQ     #IMMED2,LABEL4
         ADDX     D3,D4
         AND      LABEL5,D5
         AND      D5,LABEL5
         ANDI     #IMMED3,LABEL6
         ASL      D6,D7
         ASR      D6,D7
         ASL      #IMMED4,D0
         ASR      #IMMED4,D0
         ASL      LABEL7
JUMPBK1  ASR       LABEL7
                    C A P SEQUENCED SOFTWARE LISTING
```

```
                                C A P SEQUENCED SOFTWARE LISTING
        BLS     JUMPBK1                                                    1   51
        BGE     JUMPFW1                                                    2   52
JUMPFW1 BCHG    D1,GLOBVAR1                                                3   53
        BCHG    #BIMMED1,BLBL2                                             4   54
        BCLR    D2,BLBL3                                                   5   55
        BCLR    #BIMMED2,BLBL3                                             6   56
        BRA     BRALW           ; ALWAYS BRANCH                            7   57
BRALW   BSET    D3,BLBL4                                                   8   58
        BSET    #BIMMED3,BLBL4                                             9   59
        BSR     SUBONE          ; BRANCH TO SUBROUTINE                    10   60
        BTST    D5,BLBL5                                                  11   61
        BTST    #BIMMED4,BLBL5  ; END OF B INSTRS                         12   62
        JSR     CINST                                                     13   63
        RTS                                                               14   64
;C** ST_PROCEDURE MOD1_SIXTEENCHAR CINST                                  15   65
; ********************************************************               16   66
CINST   CHK     CLBL1,D0                                                  17   67
        CLR     CLBL2                                                     18   68
        CMP     GLOBVAR1,D1                                               19   69
        CMPA    (A1)+,A2                                                  20   70
        CMPI    #CIMMED,20(A1,D2)                                         21   71
        CMPM    (A2)+,(A3)+     ; END OF C INSTRS                         22   72
        JSR     DINST                                                     23   73
        RTS                                                               24   74
        SDBH CD1,1D                                                       25   75
;C** ST_PROCEDURE MOD1_SIXTEENCHAR DINST                                  26   76
; ********************************************************               27   77
DINST   EQU     *                                                         28   78
        DBHI    D1,CINST                                                  29   79
        DBT     D2,DINST                                                  30   80
        DIVS    20*2(A1),D3                                               31   81
        DIVU    405(PC,D1),D3   ; END OF D INSTRS                         32   82
; ********************************************************               33   83
EINST   EOR     D1,ELBL1                                                  34   84
        EORI    #$200,(A1)                                                35   85
        EXG     D3,D4                                                     36   86
        EXT     D1              ; END OF E INSTRS                         37   87
        JSR     JINST                                                     38   88
        RTS                                                               39   89
;C** ST_PROCEDURE MOD2 JINST                                              40   90
; ********************************************************               41   91
JINST   JMP     EINST           ; BACKWARD JUMP                           42   92
        JSR     SUBTWO          ; SUBROUTINE JUMP                         43   93
; ********************************************************               44   94
LINST   LEA     LINST*20(A1),A3                                           45   95
        LINK    A4,#@20         ;  @ = OCTAL                              46   96
        LSL     D1,D3                                                     47   97
        LSR     D1,D3                                                     48   98
        LSL     #LDATA,D4                                                 49   99
        LSR     GLOBVAR2(A3,D4)                                           50  100
                                C A P SEQUENCED SOFTWARE LISTING
```

C A P SEQUENCED SOFTWARE LISTING

```
         JSR      MINST                                    1  101
         RTS                                               2  102
         END                                               3  103
C** ST_PROCEDURE MOD2 MINST                                4  104
 ********************************************************  5  105
 MINST   MOVE     D1,A2                                     6  106
         MOVE     MLBL1,#GLOBVAR2                           7  107
         MOVE     #$10,SR                                   8  108
         MOVE     SR,#MLBL2                                 9  109
         MOVE     (A1)+,CCR                                 10 110
         MOVE     CCR,(A3)+                                 11 111
         MOVE     MLBL1,USP                                 12 112
         MOVE     USP,MLBL2                                 13 113
         MOVEA    MLBL4(A1),A2                              14 114
         MOVEM    D0/D1,-(SP)                               15 115
         MOVEP    D1,MLBL6(A1)                              16 116
         MOVEP    #MLBL6(A1),D1                             17 117
         MOVEQ    #-1,D1                                    18 118
         MULS     #10(A1),D5                                19 119
         MULU     MLBL5,D4          ; END OF M INSTRS       20 120
         JSR      NINSTS                                    21 121
         RTS                                               22 122
C** ST_PROCEDURE MOD2 NINSTS                               23 123
 ********************************************************  24 124
 NINSTS  EQU      *                                         25 125
         NBCD     NLBL1                                     26 126
         NEG      #HELP(D2)                                 27 127
         NEGX     D4                                        28 128
         NOP                                                29 129
         NOT      #NLBL2(D4)                                30 130
         OR       (A1)+                                     31 131
         ORI      #025,-(A2)        ; END OF N&O INSTRS     32 132
 ********************************************************  33 133
 PINST   PEA      PLBL1                                     34 134
         RESET                                             35 135
         ROL      D1,D2                                     36 136
         ROR      #2345,D4                                  37 137
         ROR      RLBL1                                     38 138
         ROXL     D3,D4                                     39 139
         ROXR     RLBL3(A1,D2)                              40 140
         RTE      THIS IS A COMMENT RTE IS A TYPE ZERO      41 141
         RTR                                               42 142
         RTS                                               43 143
                                                           44 144
C** ST_PROCEDURE MOD3 SINSTS                               45 145
 ********************************************************  46 146
 SINSTS  EQU      *                                         47 147
         SBCD     D1,D2                                     48 148
         SBCD     -(A1),-(A3)                               49 149
         SVS      SLBL4(A2)                                 50 150
```

C A P SEQUENCED SOFTWARE LISTING

C A P SEQUENCED SOFTWARE LISTING

```
        STOP    #1024                                          1  151
        SUB     SLBL.D3                                        2  152
        SUB     D4,SLBL2                                       3  153
        SUBA    SLBL2,A2                                       4  154
        SUBI    #X11,SLBL5                                     5  155
        SUBQ    #20,SLBL6                                      6  156
        SUBX    D1,D2                                          7  157
        SUBX    -(A1),-(A2)                                    8  158
        SWAP    D2                                             9  159
        JSR     TINSTS                                        10  160
        RTS                                                   11  161
;C** ST_PROCEDURE MOD3 TINSTS                                 12  162
 ********************************************************     13  163
 TINSTS EQU     *                                             14  164
        TAS     TLBL34                                        15  165
        TRN     TINSTS                                        16  166
        TRAPV   TINSTS                                        17  167
        TST     TLBL6                                         18  168
        UNLK    A1      ; END OF ALL INSTRUCTIONS             19  169
 ********************************************************     20  170
;C** ST_SKIP MOD3 TINSTS                                      21  171
        IFEQ    NONSENCE                                      22  172
        MOVE    THE,DOG                                       23  173
        ENDC                                                  24  174
;C** END_SKIP MOD3 TINSTS                                     25  175
 ********************************************************     26  176
        MOVE.L  A1,D0           ;get UCB number               27  177
        SUB.L   DCBUCB(A5),D0   ;                             28  178
        DIVU    #UCBLEN,D0      ;                             29  179
        END                                                   30  180
;C** ST_PROCEDURE MOD4 MACTESTS                               31  181
 ********************************************************     32  182
;MACTESTS  EQU *                                              33  183
; WR REG A0-A1/D0 ; WORK REGISTERS                            34  184
; *  GENERATION MACROS--                                      35  185
; SDBH MACRO                                                  36  186
;  XDEF C\1                                                   37  187
; C\1 MOVE.L (SP),-(SP) ; MAKE ROOM FOR SDB NUMB              38  188
;  MOVEM.L WR,-(SP) ; SAVE WORK REGS                          39  189
;  MOVEQ #8\2,D0 ; SET SVC NUMB                               40  190
;  ENDM                                                       41  191
;                                                             42  192
; BA MACRO                                                   43  193
;  MOVE.L -(A0),D7 ; GET ARG \1                               44  194
;  MOVE.B D7,-(A1) ; STORE ARG \1                             45  195
;  ENDM                                                       46  196
;                                                             47  197
; WA MACRO                                                   48  198
;  MOVE.L -(A0),D7 ; GET ARG \1                               49  199
;  MOVE.W D7,-(A1) ; STORE ARG \1                             50  200
```

C A P SEQUENCED SOFTWARE LISTING

C A P SEQUENCED SOFTWARE LISTING

```
 ENDM                                                                    1  201
                                                                         2  202
LA MACRO                                                                 3  203
 MOVE.L -(A0),-(A1) ; TRANSFER ARG \1                                    4  204
 ENDM                                                                    5  205
                                                                         6  206
FLR MACRO                                                                7  207
 CLR.B  (A1) ; SKIP FILLER                                               8  208
 ENDM                                                                    9  209
 PAGE                                                                   10  210
* ALLOCATE MEMORY                                                       11  211
                                                                        12  212
 SDBH ALOC.0F                                                           13  213
 LEA 16+5*4(SP),A0 ; USE A0 AS PTR TO ARGS                              14  214
 LEA 3(A0),A1 ;      A1 AS PTR TO SDB BEING BUILT                       15  215
 BA 4                                                                   16  216
 BA 3                                                                   17  217
 FLR                                                                    18  218
 LA 2                                                                   19  219
 BA 1                                                                   20  220
 BRA.L COMEND ; JOIN COMMON CODE                                        21  221
 PAGE                                                                   22  222
* COPY DISCRETES                                                        23  223
                                                                        24  224
 SDBH CDI.1D                                                            25  225
                                                                        26  226
CDI1 LEA 16+5(SP),A1 ; USE A1 AS PTR TO SDB BEING BUILT                 27  227
 BRA.L COMEND ; JOIN COMMON CODE                                        28  228
 SPC 4                                                                  29  229
* RESUME (CANCEL PAUSE)                                                 30  230
                                                                        31  231
 SDBH RESUME.07                                                         32  232
 BRA CDI1                                                               33  233
 SPC 4                                                                  34  234
* RELEASE A SEMAPHORE                                                   35  235
                                                                        36  236
 SDBH RSF.16                                                            37  237
 BRA CDI1                                                               38  238
 PAGE                                                                   39  239
* COPY EVENT FLAGS                                                      40  240
C** ST_PROCEDURE MOD4 CONDTEST                                          41  241
************************************************                        42  242
CONDTEST EQU *                                                          43  243
         IF      A1 (NE) A2 THEN                                        44  244
          ADD    D3,D4                                                  45  245
         ENDI                                                           46  246
         REPEAT                                                         47  247
          MOVE   -(A5),D2                                               48  248
         UNTIL   (EQ)                                                   49  249
         IF.B    D1 (EQ) D2 OR D3 (LT) A1 THEN                          50  250
```

C A P SEQUENCED SOFTWARE LISTING

C A P SEQUENCED SOFTWARE LISTING

```
          ADDQ.B #1,(A0)                                          1 : 251:
        ELSE                                                      2 : 252:
          MOVE     -(A5),D2                                       3 : 253:
        ENDI                                                      4 : 254:
        IF.L     D1 (EQ) D2 AND ALDO (LT) A1 THEN                 5 : 255:
          NOP                                                     6 : 256:
        ELSE                                                      7 : 257:
          ROL   D5,D2                                             8 : 258:
        ENDI                                                      9 : 259:
        FOR      D3 = #0 TO CMDLEN BY #1 DO                      10 : 260:
          MOVE.B (A3,D3),D2                                      11 : 261:
        ENDF                                                     12 : 262:
      RTS                                                        13 : 263:
END                                                              14 : 264:
```

C A P SEQUENCED SOFTWARE LISTING

C A P MODULE REPORT

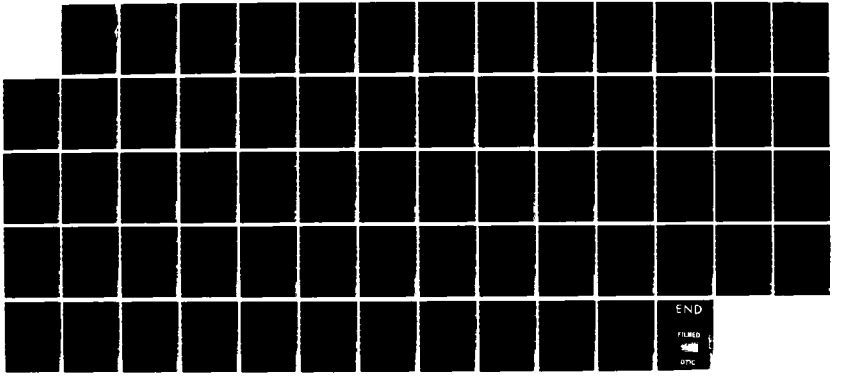
Program: FNLTEST

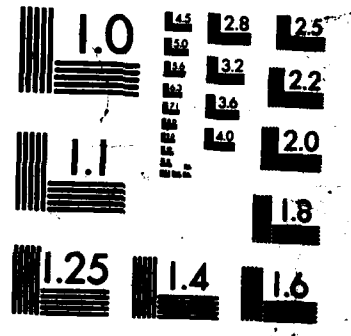
Module name	Procedure name	Page
MOD1_SIXTEENCHAR	3 Procedure(s)	
	INST_PROC_SIXTEN	1
	CINST	2
	DINST	2
MOD2	3 Procedure(s)	
	JINST	2
	MINST	3
	NINSTS	3
MOD3	2 Procedure(s)	
	SINSTS	3
	TINSTS	4
MOD4	2 Procedure(s)	
	MACTESTS	4
	CONDTEST	5

PROCEDURE NAME	ABSTRACT LINES)= 3	TOTAL LINES (=100	IN-LINE COMMENTS (=100	MAX WITH NO COMM (= 4	EXIT POINTS (= 1	BACKWARD BRANCHES)= 0	OUTWARD BRANCHES (= 0	UNCOND. BRANCHES (= 0	McCABES COMPLEX (= 10	NO. OF VARIABLE (= 50	# OF COMPLI
INST_PROC_SIXTEN	0	63	5	37	1	1	0	1	3	30	7
CINST	0	10	1	6	1	0	1	0	2	4	7
DINST	0	13	2	5	1	1	1	0	3	3	7
JINST	0	13	3	7	1	0	1	1	1	4	6
MINST	0	18	1	15	1	0	0	0	1	7	8
NINSTS	0	21	2	8	3	0	0	0	-1	5	7
SINSTS	0	16	0	16	1	0	0	0	1	6	8
TINSTS	0	13	5	4	1	1	0	1	1	6	8
MACTEST	0	59	15	5	1	2	2	4	1	11	6
CONDTEST	0	23	0	23	1	0	0	0	8	2	8

AD-A165 029 METHODOLOGY INVESTIGATION PROGRAM FLOW ANALYZER VOLUME 2/2
2(U) ARMY ELECTRONIC PROVING GROUND FORT HUACHUCA AZ
E L ANDERSON SEP 85

UNCLASSIFIED F/G 9/2 NL





COMPLIANCE TABLE

PROCEDURE NAME	ABSTRACT LINES)= 3	TOTAL LINES (=100	IN-LINE COMMENTS (=100	MAX WITH NO COMM (= 4	EXIT POINTS (= 1	BACKWARD BRANCHES)= 0	OUTWARD BRANCHES (= 0	UNCOND. BRANCHES (= 0	McCABES COMPLEX (= 10	NO. OF VARIABLE (= 50	# OF COMPLI
INST_PROC_SIXTEN	NO	YES	YES	NO	YES	YES	YES	NO	YES	YES	7
CINST	NO	YES	YES	NO	YES	YES	NO	YES	YES	YES	7
DINST	NO	YES	YES	NO	YES	YES	NO	YES	YES	YES	7
JINST	NO	YES	YES	NO	YES	YES	NO	NO	YES	YES	6
MINST	NO	YES	YES	NO	YES	YES	YES	YES	YES	YES	8
NINSTS	NO	YES	YES	NO	NO	YES	YES	YES	YES	YES	7
SINSTS	NO	YES	YES	NO	YES	YES	YES	YES	YES	YES	8
TINSTS	NO	YES	YES	YES	YES	YES	YES	NO	YES	YES	8
MACTESTS	NO	YES	YES	NO	YES	YES	NO	NO	YES	YES	6
CONDTEST	NO	YES	YES	NO	YES	YES	YES	YES	YES	YES	8

SUMMARY ITEM	ABSTRACT LINES)= 3	TOTAL LINES (=100	IN-LINE COMMENTS (=100	MAX WITH NO COMM (= 4	EXIT POINTS (= 1	BACKWARD BRANCHES)= 0	OUTWARD BRANCHES (= 0	UNCOND. BRANCHES (= 0	McCABES COMPLEX (= 10	NO. OF VARIABLE (= 50
NUMBER OF COMPLIANCES	0	10	10	1	9	10	6	6	10	10
NUMBER OF NON-COMPLIANCES	10	0	0	9	1	0	4	4	0	0
PERCENTAGE OF COMPLIANCES	0	100	100	10	90	100	60	60	100	100
PERCENTAGE OF NON-COMPLIANCES	100	0	0	90	10	0	40	40	0	0
TOTAL VALUE	0	249	34		12	5	5	7		82
AVERGE VALUE	0	25	3	13	1	1	1	1	2	8

FNLTEST SOFTWARE QUALITY METRICS REPORT

CODE ANALYSIS REPORT FOR MODULE: MOD1_SIXTEENCHAR
CODE ANALYSIS REPORT FOR PROCEDURE: INST_PROC_SIXTEN

LINES IN THE ABSTRACT	=	3
COMMENT ONLY LINES	=	0
EXECUTABLE LINES	=	31
NON-EXECUTABLE LINES	=	22
LINES SKIPPED	=	0
COMMENTS FOR EXECUTABLE LINES	=	N/A
COMMENTS FOR NON-EXECUTABLE LINES	=	N/A
COMMENTS ON A LINE OF CODE	=	5
PERCENT OF EXECUTABLE LINES COMMENTED	=	N/A
PERCENT OF NON-EXECUTABLE LINES COMMENTED	=	N/A
MAXIMUM CONSECUTIVE LINES WITHOUT A COMMENT	=	37
EXECUTABLE STATEMENTS	=	31
NON-EXECUTABLE STATEMENTS	=	22
MULTI-LINE STATEMENTS	=	N/A
MULTI-STATEMENT LINES	=	N/A
LINES OF CODE (PRIMARY LANGUAGE)	=	31
LINES OF CODE (EMBEDDED LANGUAGE)	=	N/A
PRIMARY VS. EMBEDDED LANGUAGE SWITCHES	=	N/A
NUMBER OF VARIABLES	=	23
TOTAL LINES	=	56
ENTRY POINTS	=	4
EXIT POINTS	=	1
INCLUDE STATEMENTS	=	2
FORWARD BRANCHES	=	2
BACKWARD BRANCHES	=	1
BRANCHES OUT OF THE ROUTINE	=	0
CONDITIONAL BRANCHES	=	2
UNCONDITIONAL BRANCHES	=	1

METRICS REPORT FOR PROCEDURE: INST_PROC_SIXTEN

NODES	=	8
PATHS	=	9
MCCABE'S CYCLOMATIC	=	3

HALSTEAD'S ANALYSIS :

UNIQUE OPERATORS	=	23
UNIQUE OPERANDS	=	31
TOTAL OPERATOR USAGE	=	87
TOTAL OPERAND USAGE	=	48
PROGRAM LENGTH	=	135
PROGRAM VOLUME	=	776.910
PROGRAM LEVEL	=	0.056
POTENTIAL VOLUME	=	43.631
DIFFICULTY	=	17.806
EFFORT	=	13834.008

FNLTEST SOFTWARE QUALITY METRICS REPORT

CODE ANALYSIS REPORT FOR MODULE: MOD1_SIXTEENCHAR
CODE ANALYSIS REPORT FOR PROCEDURE: CINST

LINES IN THE ABSTRACT	=	0
COMMENT ONLY LINES	=	0
EXECUTABLE LINES	=	9
NON-EXECUTABLE LINES	=	0
LINES SKIPPED	=	0
COMMENTS FOR EXECUTABLE LINES	=	N/A
COMMENTS FOR NON-EXECUTABLE LINES	=	N/A
COMMENTS ON A LINE OF CODE	=	1
PERCENT OF EXECUTABLE LINES COMMENTED		= N/A
PERCENT OF NON-EXECUTABLE LINES COMMENTED		= N/A
MAXIMUM CONSECUTIVE LINES WITHOUT A COMMENT	=	6
EXECUTABLE STATEMENTS	=	9
NON-EXECUTABLE STATEMENTS	=	0
MULTI-LINE STATEMENTS	=	N/A
MULTI-STATEMENT LINES	=	N/A
LINES OF CODE (PRIMARY LANGUAGE)	=	9
LINES OF CODE (EMBEDDED LANGUAGE)	=	N/A
PRIMARY VS. EMBEDDED LANGUAGE SWITCHES	=	N/A
NUMBER OF VARIABLES	=	3
TOTAL LINES	=	9
ENTRY POINTS	=	1
EXIT POINTS	=	1
INCLUDE STATEMENTS	=	0
FORWARD BRANCHES	=	0
BACKWARD BRANCHES	=	0
BRANCHES OUT OF THE ROUTINE	=	1
CONDITIONAL BRANCHES	=	1
UNCONDITIONAL BRANCHES	=	0

METRICS REPORT FOR PROCEDURE: CINST

NODES	=	3
PATHS	=	3
MCCABE'S CYCLOMATIC	=	2

HALSTEAD'S ANALYSIS :

UNIQUE OPERATORS	=	12
UNIQUE OPERANDS	=	12
TOTAL OPERATOR USAGE	=	30
TOTAL OPERAND USAGE	=	14
PROGRAM LENGTH	=	44
PROGRAM VOLUME	=	201.738
PROGRAM LEVEL	=	0.143
POTENTIAL VOLUME	=	28.820
DIFFICULTY	=	7.000
EFFORT	=	1412.168

FNLTEST SOFTWARE QUALITY METRICS REPORT

CODE ANALYSIS REPORT FOR MODULE: MOD1_SIXTEENCHAR
CODE ANALYSIS REPORT FOR PROCEDURE: DINST

LINES IN THE ABSTRACT	=	0
COMMENT ONLY LINES	=	0
EXECUTABLE LINES	=	10
NON-EXECUTABLE LINES	=	1
LINES SKIPPED	=	0
COMMENTS FOR EXECUTABLE LINES	=	N/A
COMMENTS FOR NON-EXECUTABLE LINES	=	N/A
COMMENTS ON A LINE OF CODE	=	2
PERCENT OF EXECUTABLE LINES COMMENTED	=	N/A
PERCENT OF NON-EXECUTABLE LINES COMMENTED	=	N/A
MAXIMUM CONSECUTIVE LINES WITHOUT A COMMENT	=	5
EXECUTABLE STATEMENTS	=	10
NON-EXECUTABLE STATEMENTS	=	1
MULTI-LINE STATEMENTS	=	N/A
MULTI-STATEMENT LINES	=	N/A
LINES OF CODE (PRIMARY LANGUAGE)	=	10
LINES OF CODE (EMBEDDED LANGUAGE)	=	N/A
PRIMARY VS. EMBEDDED LANGUAGE SWITCHES	=	N/A
NUMBER OF VARIABLES	=	1
TOTAL LINES	=	11
ENTRY POINTS	=	2
EXIT POINTS	=	1
INCLUDE STATEMENTS	=	0
FORWARD BRANCHES	=	0
BACKWARD BRANCHES	=	1
BRANCHES OUT OF THE ROUTINE	=	1
CONDITIONAL BRANCHES	=	2
UNCONDITIONAL BRANCHES	=	0

METRICS REPORT FOR PROCEDURE: DINST

NODES	=	5
PATHS	=	6
MCCABE'S CYCLOMATIC	=	3

HALSTEAD'S ANALYSIS :

UNIQUE OPERATORS	=	13
UNIQUE OPERANDS	=	12
TOTAL OPERATOR USAGE	=	30
TOTAL OPERAND USAGE	=	17
PROGRAM LENGTH	=	47
PROGRAM VOLUME	=	218.261
PROGRAM LEVEL	=	0.109
POTENTIAL VOLUME	=	23.703
DIFFICULTY	=	9.208
EFFORT	=	2009.822

FNLTEST SOFTWARE QUALITY METRICS REPORT

GLOBAL CODE ANALYSIS REPORT FOR MODULE: MOD1_SIXTEENCHAR

LINES IN THE ABSTRACT	=	3
COMMENT ONLY LINES	=	0
EXECUTABLE LINES	=	50
NON-EXECUTABLE LINES	=	23
LINES SKIPPED	=	0
COMMENTS FOR EXECUTABLE LINES	=	N/A
COMMENTS FOR NON-EXECUTABLE LINES	=	N/A
COMMENTS ON A LINE OF CODE	=	8
PERCENT OF EXECUTABLE LINES COMMENTED	=	N/A
PERCENT OF NON-EXECUTABLE LINES COMMENTED	=	N/A
MAXIMUM CONSECUTIVE LINES WITHOUT A COMMENT	=	37
EXECUTABLE STATEMENTS	=	50
NON-EXECUTABLE STATEMENTS	=	23
MULTI-LINE STATEMENTS	=	N/A
MULTI-STATEMENT LINES	=	N/A
LINES OF CODE (PRIMARY LANGUAGE)	=	50
LINES OF CODE (EMBEDDED LANGUAGE)	=	N/A
PRIMARY VS. EMBEDDED LANGUAGE SWITCHES	=	N/A
NUMBER OF VARIABLES	=	27
TOTAL LINES	=	76
ENTRY POINTS	=	7
EXIT POINTS	=	3
INCLUDE STATEMENTS	=	2
FORWARD BRANCHES	=	2
BACKWARD BRANCHES	=	2
BRANCHES OUT OF THE ROUTINE	=	2
CONDITIONAL BRANCHES	=	5
UNCONDITIONAL BRANCHES	=	1

METRICS REPORT FOR PROCEDURE: GLOBAL

NODES	=	16
PATHS	=	18
MCCABE'S CYCLOMATIC	=	8

HALSTEAD'S ANALYSIS IS NOT APPLICABLE:

C A P CALLS REPORT

Program: FNLTEST

Procedure name	Procedure called
INST_PROC_SIXTEN	2 Procedure(s) called
	SUBONE CINST
CINST	1 Procedure(s) called
	DINST
DINST	1 Procedure(s) called
	JINST
JINST	2 Procedure(s) called
	SUBTWO MINST
MINST	1 Procedure(s) called
	NINSTS
NINSTS	0 Procedure(s) called
SINSTS	1 Procedure(s) called
	TINSTS
TINSTS	0 Procedure(s) called
MACTESTS	0 Procedure(s) called

C A P CALLS REPORT

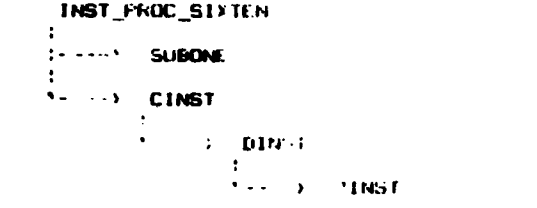
Program: FNLTEST

Procedure name	Procedure called
CONDTEST	0 Procedure(s) called

C A P STRUCTURE CHART FOR PROGRAM: FNLTEST
MODULE: MOD1_SIXTEENCHAR
PROCEDURE: INST_PROC_SIXTEN

LEVEL OF CALL

1	2	3	4	5	6	7	8	9	10	11	12	13	14



UNDEFINED EXTERNALS REPORT

SUBROUTINE CALLS

SUBONE
SUBTWO

GLOBAL VARIABLE USAGE REPORT
PROGRAM: FNLTEST
MODULE: MOD3

VARIABLE NAME	TYPE	NUMBER OF TIMES				PROCEDURE
		MOD.	REF.	TEST	ARG.	DEF.	

*** NO GLOBAL VARIABLES TO REPORT FOR THIS MODULE ***

VARIABLE USAGE REPORT

PROGRAM: FNLTEST
MODULE: MOD3
PROCEDURE: SINSTS

VARIABLE NAME	TYPE	NUMBER OF TIMES MOD.	REF.	TEST	ARG.	DEF.
SLBL		0	1	0	0	0
SLBL2		1	1	0	0	0
SLBL4		0	0	0	0	1
SLBL5		1	0	0	0	0
SLBL6		1	0	0	0	0
LOCAL REPORT TOTALS						
5		3	2	0	0	1

0

VARIABLE USAGE REPORT
PROGRAM: FNLTEST
MODULE: MOD3
PROCEDURE: TINSTS

VARIABLE NAME	TYPE	NUMBER OF TIMES MOD.	REF.	TEST	ARG.	DEF.
DCBUCB		0	1	0	0	0
TLBL34		1	0	0	0	0
UCBLEN		0	1	0	0	0
LOCAL REPORT TOTALS						
3		1	2	0	0	0

MACRO USAGE REPORT
PROGRAM: FNLTEST
LANGUAGE M68000

DEFINITION				USAGE	
MACRO NAME	DEFINING MODULE	PAGE NUMBER	LENGTH (LINES)	PROCEDURE	NUMBER OF OCCURANCES
BA	MOD4	4	4	MACTESTS	3
FLR	MOD4	5	3	MACTESTS	1
LA	MOD4	5	3	MACTESTS	1
SDBH	MOD4	4	6	CINST MACTESTS	1 4
WA	MOD4	4	4	*** THIS MACRO IS NOT USED ***	

TOTALS :

NUMBER OF MACROS USED = 5
NUMBER OF UNDEFINED MACROS = 0
NUMBER OF UN-USED MACROS = 1

APPENDIX A
MASTER FILE DESCRIPTION

The master file contains a concatenation of all of the principle files resulting from the ACAP static analysis. This file has a user-specified file name and is in namelist format. For more information on namelist-directed I/O, see the VAX-11 Fortran Language Reference Manual.

A.1 NAMELIST - DIRECTED I/O

As specified in the VAX-11 Fortran Reference Manual, each namelist data block has the form:

$group-name entity = value [,entity = value,...]$[END]

where:

$ is the special symbol used to indicate the beginning or end of input. The ampersand (&) can be used in place of the dollar sign.

group-name is the name of the namelist that contains the entity or entities to be given values. The namelist must have been previously defined in a NAMELIST statement in the program unit.

entity is a namelist-defined:

- Variable
- Array Name
- Subscripted Variable
- Variable With a Substring
- Subscripted Variable With a Substring

value is a constant, a list of constants, a repetition of constants in the form r*c, or a repetition of values in the form r* value.

END is an optional part of the last delimiter.

A.2 MASTER FILE FORMAT

The five files from which the master file is constructed are:

VUR.NML - The file containing the data on variable usage.

BRANCHES.NML - The file containing the data on branches.

SQM.NML - The file containing the data on software quality metrics.

MACRO.NML - The file containing the data on MACROS.

CONTROL.NML - The ACAP control file.

ERROR.NML - The file containing the errors accumulated during the ACAP run.

At the beginning of each of these files is the data block for the namelist group-name "ID". This data is followed by the data for the other namelist group-names in that file. Each of these files contains a variable number of namelist data blocks dependent on the source code being analyzed and the user options selected.

The format of the master file can be summarized as follows:

File Name	Group-Name	Number
VUR.NML	ID	There is one per VUR.NML file.
	VAR_LISTS	There is one VAR_LISTS for each module followed by all the VARUSAGE groups for that module.
	VARUSAGE	There is one VARUSAGE group for each procedure plus one for the global variables in the module. The "global" VARUSAGE is flagged by PROC_SEQ = 0 and must precede the "procedure" VARUSAGE groups.

File Name	Group-Name	Number
BRANCHES.NML	ID	There is one per BRANCHES.NML file.
	PROC_LISTS	There is one PROC_LISTS for each module followed by all the CALLS groups for that module.
	CALLS	One per procedure.
SQM.NML	ID	There is one per SQM.NML file.
	METRICS	There is one METRICS group for each procedure in a module plus one for the entire module. The METRICS group for the entire module follows the others and is flagged by PROC_NAME = 'GLOBAL' and PROC_SEQ = 0.
MACRO.NML	MACS	One per MACRO occurrence.
CONTROL.NML	ID	1 per CONTROL.NML file.
	PARSER_TABLE	1 per CONTROL.NML file.
ERROR.NML	ID	1 per Group-Name
	SCANNER_ERR	1 per error from SCANNER.
	ID	1 per Group-Name
	BRANCHES_ERR	1 per error from BRANCHES.
	ID	1 per Group-Name
	VARUSE_ERR	1 per error from VARUSE.
	ID	1 per Group-Name
	SQMETS_ERR	1 per error from SQMETS.

A.3 NAMELIST GROUP DESCRIPTIONS

The entities within each of these namelist group-names are described in the following table. There is a type specified for each entity which takes the form:

TYPE N

where:

TYPE is either Char for CHARACTER data; Int for INTEGER data

N is the number of bytes.

The Dimensions column gives the size of the array dimensions if the variable is an array.

TABLE - NAMELIST DESCRIPTIONS

Entity	Type	(Dimensions)	Description
ID			
PROGRAM_NAME	Char 15		The name of the program.
SOURCE_FILE	Char 15		Source file being analyzed.
SOURCE_FILE_CLASS	Char 1		Classification of source file: U = Unclassified C = Confidential S = Secret T = Top Secret Any Other Character = ERROR
SOURCE_LANGUAGE	Char 15		Source language selected.
DATE_TRNSL	Char 10		Date translation of source occurred.
TIME_TRNSL	Char 10		Time translation of source occurred.
FILE_NAME	Char 15		'SQM.NML', 'VUR.NML', BRANCHES.NML', or 'CONTROL,NML'; or for ERROR.NML file, **Proc Name** where Proc Name is SCANNER, VARUSE, BRANCHES, OR SQMETS.
VAR_LISTS			There is one VAR_LISTS for each module followed by all the VARUSAGE groups for that module.
MOD_NAME	Char 31		Module name.
N_PROCS	Int 4		Number of procedures in module (Max = 400).
PROC_KEY	Int 4	(400)	Variable flag for each procedure: 0 = Contains no variables, Else = Number of variables.
PROC_LIST	Char 31	(400)	Names of procedures.
VAR_LIST	Char 31	(3000)	Names of variables.

TABLE - NAMELIST DESCRIPTIONS (Continued)

Entity	Type	(Dimensions)	Description
VARUSAGE			There is one VARUSAGE group for each procedure plus one for the global variables in the module. The "global" VARUSAGE is flagged by PROC_SEQ = 0 and must precede the "procedure" VARUSAGE groups.
"Global" VARUSAGE			The format for the "global" VARUSAGE group is as follows:
PROC_SEQ	Int 4		Procedure sequence number in module. 0 for "Global".
L_VARS	Int 4		Number of entries in VAR_USAGE. There is an entry for the usage of each global variable in each procedure in which it appears.
VAR_USAGE	Int 4	(6,3000)	Variable usage for each procedure, N = 1 to L_VARS. (1,N) Index to the global variable name in VAR_LIST. (2,N) Number of times modified. (3,N) Number of times referenced. (4,N) Number of time tested. (5,N) Number of times variable is used as an argument. (6,N) Number of times defined.
VAR_TYPE	Char 3	(3000)	User defined variable type printed on Variable Usage Report.
VAR_ACCESS	Int 4	(3000)	Index to procedure name in PROC_LIST (contained in PROC_LIST group). The global variable named in VAR_USAGE (1,N) above appears in the procedure named by VAR_ACCESS (N) with the variable usage as specified in VAR_USAGE.
"Procedure" VARUSAGE			The format for the "procedure" VARUSAGE group is as follows:
PROC_SEQ	Int 4		Procedure sequence number in module.

TABLE - NAMELIST DESCRIPTIONS (Continued)

Entity	Type	(Dimensions)	Description
L_VARS	Int 4		Number of variables in this procedure.
VAR_USAGE	Int 4	(6,3000)	Variable usage for this procedure, N = 1 to L_VARS. (1,N) Index to the variable name in VAR_LIST. (2,N) Number of times modified. (3,N) Number of times referenced. (4,N) Number of time tested. (5,N) Number of times variable is used as an argument. (6,N) Number of times defined.
VAR_TYPE	Char 3	(3000)	User defined variable type printed on Variable Usage Report.
VAR_ACCESS	Int 4	(3000)	Data access information. 0 - Local Variable 1 - Global Variable 2 - Argument
PROC_LISTS			There is one PROC_LISTS for each module followed by all the CALLS groups for that module.
MOD_NAME	Char 31		Module name.
N_PROC	Int 4		Number of procedures in module (Max = 400).
PROC_LIST	Char 31	(400)	Names of procedures.
PROC_LINE	Int 4	(3000)	Line number of procedure.
N_ENTRY_POINTS	Int 4		Number of entry points.
ENTRY_POINT_LIST	Char 31	(3000)	Names of the entry points.
CALLS			One per procedure.
PROC_SEQ	Int 4		Sequence number in module.
NUM_CALLS	Int 4		Number of procedure calls.
PROC_CALLS	Char 31	(400)	Procedure names.

TABLE - NAMELIST DESCRIPTIONS (Continued)

Entity	Type	(Dimensions)	Description
PROC_CALLS_LN	Int 4	(400)	The line number from which the call was made.
METRICS			There is one METRICS group for each procedure in a module plus one for the entire modoule. The METRICS group for the entire module follows the others and is flagged by PROC_NAME = 'GLOBAL' and PROC_SEQ = 0.
MOD_NAME	Char 31		Name of module.
MOD_SEQ	Int 4		Sequence number of module.
PROC_NAME	Char 31		Name of procedure.
PROC_SEQ	Int 4		Sequence number of procedure.
N_ABSTRACT	Int 4		Number of lines in the abstract.
N_COMMENT_LINES	Int 4		Number of comment only lines.
N_LINES_X	Int 4		Number of lines where an executable is the first instruction on the line.
N_LINES_NX	Int 4		Number of lines where non-executable is the first instruction on the line.
N_LINES	Int 4		Total number of lines.
N_COMMENTS_X	Int 4		Number of comments in executable statements.
N_COMMENTS_NX	Int 4		Number of comments in non-executable statements.
N_INLINE_COMMENTS	Int 4		Number of lines of code which contain a comment.
N_PER_COMM_X	Int 4		Percent of executable lines commented.
N_PER_COMM_NX	Int 4		Percent of non-executable lines commented.

TABLE - NAMELIST DESCRIPTIONS (Continued)

Entity	Type	(Dimensions)	Description
MAX_WO_COMMENTS	Int 4		Maximum number of consecutive lines without comments.
N_STAMNTS_X	Int 4		Number of executable statements.
N_STAMNTS_NX	Int 4		Number of non-executable statements.
MULT_LINE_STAMNTS	Int 4		Number of compound lines.
MULT_STAMNT_LINES	Int 4		Number of multi statement lines.
N_ENTRY_PTS	Int 4		Number of entry points.
N_EXIT_PTS	Int 4		Number of exit points.
N_BRANCH_FRWD	Int 4		Number of forward branches.
N_BRANCH_BACK	Int 4		Number of backward branches.
N_BRANCH_OUT	Int 4		Number of branches out of the routine.
N_BRANCH_COND	Int 4		Number of conditional branches.
N_BRANCH_UNCOND	Int 4		Number of unconditional branches.
N_NODES	Int 4		Number of nodes, 0 if no executable statements.
N_PATHS	Int 4		Number of paths, 0 if no executable statements.
MCCABES_CYCLOMATIC	Int 4		McCabe's cyclomatic, 0 is no executable statements, else paths - nodes + 2.
N_LINES_SKIPPED	Int 4		Number of lines that were skipped by the parser (ST_SKIP .. END_SKIP).
N_INCLUDES	Int 4		Number of include statements.
N_OPERATORS	Int 4		Number of unique operators.
N_OPERATORS_USED	Int 4		Number of operators used.
N_OPERANDS	Int 4		Number of unique operands.

TABLE - NAMELIST DESCRIPTIONS (Continued)

Entity	Type	(Dimensions)	Description
N_OPERANDS_USED	Int 4		Number of operands used.
N_PRIM_LINES	Int 4		Number of lines of primary code.
N_EMBED_LINES	Int 4		Number of lines of embedded code.
N_SWITCH	Int 4		Number of switches from primary to embedded code and back.
N_VARS	Int 4		Number of variables.
CODE_COUNTS	Int 4	20	Instruction code usage in a procedure.
MACS			
MAC_NAME	Char 16		MACRO name.
DEF_USE_FLAG	Char 1		Flag indicating whether occurance is a definition or use (D or U).
PROC_NAME	Char 16		Defining module or using procedure name.
N_USE	Int 4		If DEF_USE_FLAG = D N_USE is page number of definition else it is the number of times used in a procedure.
MAC_SIZE	Int 4		MACRO size in lines.
PARSER TABLE			
INSTRUMENT_VALUE	Char 15		User-defined ASCII string that is not a valid construct of target language used to identify instrumentation lines.
PARSER_PRNT	Int 4		Flag used to turn on output to debug file PARSER.FIL.
STRINGS_PRNT	Int 4		Flag used to turn on output to debug file STRINGS.FIL.
COMMENT_PRNT	Int 4		Flag used to turn on output to debug file COMMENT.FIL.
INSTRM_PRNT	Int 4		Flag used to turn on output to debug file INSTRM.FIL.

TABLE - NAMELIST DESCRIPTIONS (Continued)

Entity	Type	(Dimensions)	Description
N_ASSM	Int 4		Number of assembly language mnemonics.
ASSM_INSTS	Char 8	(200)	Assembly language mnemonics.
ASSM_CODES	Int 4	(200)	Code for each mnemonic.
ASSM_MODES	Int 4	(200)	Address mode for each mnemonic.
ASSM_SPECS	Int 4	(200)	Special information about the instruction.
N_USER	Int 4		Number of user-defined instructions.
USER_INSTS	Char 8	(100)	User mnemonics.
USER_CODES	Int 4	(100)	Code for each user instruction.
USER_MODES	Int 4	(100)	Address mode for each user instruction.
USER_SPECS	Int 4	(100)	Special information about each user instruction.
SCANNER_ERR			
ERROR_CODE	Char 3		Error code. ML = Multi-line error. IRM = Incorrect instrumentation construct or non-comment in abstract.
MOD_NAME	Char 31		Module name.
PROC_NAME	Char 31		Procedure name.
N_LINES	Int 4		Line number in the procedure.
LINE	Char 80		Line being scanned.
BRANCHES_ERR			
MOD_NAME	Char 31		Name of module.
PROC_NAME	Char 31		Procedure name.
ERROR_CODE	Char 3		Error code (none currently defined).

TABLE - NAMELIST DESCRIPTIONS (Continued)

Entity	Type	(Dimensions)	Description
N_LINES	Int 4		Line number in the procedure.
VARUSE_ERR			
MOD_NAME	Char 31		Module name.
ERROR_CODE	Char 3		Error code: SRT = Sort failed
SQMETS_ERR			
ERROR_CODE	Char 3		Error code: NML = PROC names in namelists do not match. GEN = Invalid assembly language code.
MOD_NAME	Char 31		Module name.
PROC_NAME	Char 31		Procedure name.
N_LINES	Int 4		Line number in the procedure.
LINE	Char 80		Line being scanned.

APPENDIX B
SOURCE CODE INSTRUMENTATION

In ACAP, the source code is grouped into modules and procedures for analysis. A *procedure* is the smallest *named grouping* of source code. A module consists of multiple procedures. All of the metrics extracted by ACAP are reported at the procedure and the module level.

Instrumentation is used to identify the procedures and modules in the assembly language source code being analyzed. In addition, instrumentation can be used to identify abstracts (or prologues) and to skip lines. Instrumentation may also serve a useful side benefit as a mechanism by which the analyst can tie a documented view of a software system's procedures to the actual source code.

Five types of instrumentation are available to the user:

```
*** ST_PROCEDURE  "module name"  "procedure name"
*** ST_ABSTRACT  "module name"  "procedure name"
*** END_ABSTRACT
*** ST_SKIP
*** END_SKIP
```

where *** is a user defined ASCII string of up to 15 characters in length (see Section 3.1.2) chosen so as to not be a valid construct of the target language. "Module Name" is a string identifying the module, "Procedure Name" is a string identifying the procedure. These strings may vary in size depending on the language to be analyzed, currently the M68000 and 6800 languages allow these strings to be 16 characters long.

An example of the use of instrumentation is as follows:

```
C** ST_ABSTRACT MOD1 PROC1
* PROC1 - SAMPLE ABSTRACT FOR PROC1
o
o
o
*
C** END_ABSTRACT
C** ST_PROCEDURE MOD1 PROC1
START    EQU   *
         LDA A  VAR
          o
          o
          o
         END
C** ST_SKIP
         NAM 'SKIP ASSEMBLER DIRECTIVES'
         OPT   PAG
C** END_SKIP
C** ST_PROCEDURE MOD1 PROC2
 o
 o
 o
```

This example defines an abstract preceding a procedure and skips some lines of assembly diretives at the end of the procedure.

B.1 PROCEDURE INSTRUMENTATION

The minimum instrumentation that a user of ACAP must utilize is the ST_PROCEDURE line. ACAP recognizes this line as the start of a new procedure and the end of the previous procedure (the first and last procedures in a file are handled within their respective contexts). All comment lines between this line and the first instruction in the procedure are, by default, reported as belonging to the abstract unless the abstract instrumentation is used.

B.2 ABSTRACT

Where appropriate, the user may override the default processing for abstracts by utilizing the ST_ABSTRACT and END_ABSTRACT instrumentation lines. These will cause ACAP to associate the user specified abstract to a specific procedure. ACAP then reports the total number of lines in this user defined abstract. In addition, any executable and non-executable statement lines (i.e., non-comments) found within the abstract are reported in the ERROR.NML.

B.3 SKIP

The user may utilize the ST_SKIP and END_SKIP instrumentation lines to "skip" over code within a procedure. Skip instrumentation might be used to avoid ambiguities arising from lines which are conditionally assembled. ACAP reports the number of lines skipped during processing.

Note: ACAP will not handle the case where a subroutine is instrumented within another instrumented routine. Therefore, this practice should be avoided, the following is an example of an illegal instrumentation:

```
C** ST_PROCEDURE MOD1 PROC1
START   EQU *
        LDA A VAR
        o
        o
C** ST_PROCEDURE MOD1 PROC2
        SUB A #10
        o
        o
        o
C** ST_PROCEDURE MOD1 PROC1
        o
        o
        o
```

The occurrence of a "ST_PROCEDURE" automatically ends the collection of metrics for the previous procedure and the previous procedure may not be repeated or added to.

APPENDIX C
MODIFICATIONS NECESSARY TO ADD A LANGUAGE TO ACAP

C.1 DEVELOPMENT OF NEW SOFTWARE ROUTINES

In order to conform with the current design of ACAP the following language specific programs need to be developed for each language:

1. Scanner: This program prepares lines of instrumented source code for parsing and accumulates various procedure oriented metrics.

2. Parser: This program breaks down an instrumented source line into identifiable language independant tokens and, for applications requiring Halsteads Analysis, collects operator and operand information. In order to accomodate the identification of tokens a "Parser Table" must be developed which contains information about every instruction in the the target language.

3. Parser Table: This is a non-executable file with language specific information stored in namelist format. Each Parser Table must be identifiable by a unique name. At the time of language selection the appropriate parser table is copied into a file with the generic name "PARSER.TBL" and is used in various programs until another language selection is made. See Appendix D "ACAP Language Capabilities" for more specific information on the parser tables.

The end result of the above programs is the file "TRNSL.FIL". This file contains generic information about the language being analyzed and is the input to the static analyzer programs which are, and should remain, language independant.

C.2 MODIFICATION OF EXISTING SOFTWARE ROUTINES

These changes are basically to add the new language to the menu's so it can be selected by an operator.

1. ASMCHOICE.MEN, ACAPCAP.MEN: Both of these menu's need to have the new language name added.

2. ASMCHOICE.COM: This program determines which "scanner" and "Parser Table" are to be used in subsequent processing and puts the source language name in the control file. Therefore this program needs modification to add this information for any new language.

APPENDIX D
ACAP LANGUAGE CAPABILITIES

D.1 6800

1. 6800 PARSER TABLE: The following describes the Parser Table (P6800.TBL) used to analyze 6800 programs:

```
CHARACTER*15  INSTRUMENT_VALUE
INTEGER       N_USER, N_ASSM
INTEGER*4     ASSM_CODES(200), ASSM_MODES(200)
INTEGER*4     ASSM_SPECS(200), USER_CODES(100)
INTEGER*4     USER_MODES(100), USER_SPECS(100)
INTEGER*4         PARSER_PRNT,  COMMENT_PRNT,  STRINGS_PRNT,
              INSTRUM_PRNT

 NAMELIST/PARSER_TABLE/INSTRUMENT_VALUE
*                     ,PARSER_PRNT
*                     ,STRINGS_PRNT
*                     ,COMMENT_PRNT
*                     ,INSTRM_PRNT
*                     ,N_ASSM
*                     ,ASSM_INSTS
*                     ,ASSM_CODES
*                     ,ASSM_MODES
*                     ,ASSM_SPECS
*                     ,N_USER
*                     ,USER_INSTS
*                     ,USER_CODES
*                     ,USER_MODES
*                     ,USER_SPECS
```

Where:

N_ASSM = Number of assembly language constructs defined in ASSM_INSTS that the parser is to recognize. These are both instructions and assembler directives.

ASSM_INSTS = The actual assembly language mnemonics and assembler directives. These can be up to eight characters.

ASSM_CODE = Code assigned to each entry in ASSM_INSTS which is used by the Static Analyzer. The following codes have been defined:

-1 = Undefined
1 = Modifies Argument
2 = References Argument
3 = Tests Argument
4 = Performs Work
5 = Branches Conditionally
6 = Branches Unconditionally
7 = Calls Subroutine
8 = Returns From Subroutine
9 = Defines Argument
11 = Declares Variable
12 = Declares Constant
13 = Defines Constant
14 = Performs Non-Executable Work

ASM_MODE = Addressing mode or more specifically the number of operands that can follow a particular instruction. Basically, there are three modes:

0 = No Operands
1 = One Operands
2 = Two Operands

ASSM_SPECS = Special information about an instruction (not used for 6800 language).

2. 6800 NON-APPLICABLE METRICS: Also contained in the "Parser Table" is a namelist which contains any metrics which are predetermined to be non-applicable. The format is as follows:

```
      NAMELIST/INIT_MET/ N_COMMENT_LINES
     *                  ,N_LINES_X
     *                  ,N_LINES_NX
     *                  ,N_LINES
     *                  ,N_COMMENTS_X
     *                  ,N_COMMENTS_NX
     *                  ,N_INLINE_COMMENTS
     *                  ,N_PER_COMM_X
     *                  ,N_PER_COMM_NX
     *                  ,MAX_WO_COMMENTS
     *                  ,N_STAMNTS_X
     *                  ,N_STAMNTS_NX
     *                  ,MULT_LINE_STAMNTS
     *                  ,MULT_STAMNT_LINES
     *                  ,N_ENTRY_PTS
     *                  ,N_BRANCH_FRWD
     *                  ,N_BRANCH_BACK
     *                  ,N_BRANCH_OUT
     *                  ,N_BRANCH_COND
     *                  ,N_BRANCH_UNCOND
     *                  ,N_NODES
     *                  ,N_PATHS
```

```
*                   ,MCCABES_CYCLOMATIC
*                   ,N_LINES_SKIPPED
*                   ,N_INCLUDES
*                   ,N_OPERATORS
*                   ,N_OPERATORS_USED
*                   ,N_OPERANDS
*                   ,N_OPERANDS_USED
*                   ,N_PRIM_LINES
*                   ,N_EMBED_LINES
*                   ,N_SWITCH
*                   ,N_VARS
*                   ,CODE_COUNTS
```

This entry in the parser table should contain ONLY the metrics which are not applicable. For the 6800 language the following are set to -1 indicating non-applicability:

```
N_COMMENTS_X
N_COMMENTS_NX
N_PER_COMM_X
N_PER_COMM_NX
MULT_LINE_STAMNTS
MULT_STAMNTS_LINES
N_INCLUDES
N_OPERATORS
N_OPERATORS_USED
N_OPERANDS
N_OPERANDS_USED
N_PRIM_LINES
N_EMBED_LINES
```

3. 6800 LANGUAGE ANALYSIS LIMITATIONS:

    1. Halsteads parameters (number of unique operators, number of operator occurrences, number of unique operands, and number of operand occurrences) are not collected.

    2. Variable usage data relating more to higher level language, such as variable type and number of times a variable is passed as an argument, is not collected.

Table D-1 shows the content of the Parser Table for the 6800 language.

TABLE D-1

6800 PARSER TABLE

	ASSM-INSTS	ASSM-CODES	ASSM-MODES	Description
1	ABA	4	0	Add Accumulators
2	ADC	2	2	Add With Carry
3	ADD	2	2	Add
4	AND	2	2	Logical And
5	ASL	4	1	Arithmetic Shift Left
6	ASR	4	1	Arithmetic Shift Right
7	BCC	5	1	Branch if Carry Clear
8	BCS	5	1	Branch if Carry Set
9	BEQ	5	1	Branch if Equal to Zero
10	BGE	5	1	Branch if Greater or Equal Zero
11	BGT	5	1	Branch if Greater Than Zero
12	BHI	5	1	Branch if Higher
13	BIT	2	2	Bit Test
14	BLE	5	1	Branch if Less or Equal
15	BLS	5	1	Branch if Lower or Same
16	BLT	5	1	Branch if Less Than Zero
17	BMI	5	1	Branch if Minus
18	BNE	5	1	Branch if Not Equal to Zero
19	BPL	5	1	Branch if Plus
20	BRA	6	1	Branch Always
21	BSR	7	1	Branch to Subroutine
22	BVC	5	1	Branch if Overflow Clear
23	BVS	5	1	Branch if Overflow Set
24	CBA	4	0	Compare Accumulators
25	CLC	4	0	Clear Carry
26	CLI	4	0	Clear Interrupt Mask
27	CLR	1	1	Clear
28	CLV	4	0	Clear Overflow
29	CMP	2	2	Compare
30	COM	1	1	Complement
31	CPX	2	1	Compare Index Register

TABLE D-1 (Continued)
6800 PARSER TABLE

	ASSM-INSTS	ASSM-CODES	ASSM-MODES	Description
32	DAA	4	0	Decimal Adjust
33	DEC	1	1	Decrement
34	DES	4	0	Decrement Stack Pointer
35	DEX	4	0	Decrement Index Register
36	EOR	2	2	Exclusive OR
37	INC	1	1	Increment
38	INS	4	0	Increment Stack Pointer
39	INX	4	0	Increment Index Register
40	JMP	6	1	Jump
41	JSR	7	1	Jump to Subroutine
42	LDA	2	2	Load Accumulator
43	LDS	2	1	Load Stack Pointer
44	LDX	2	1	Load Index Register
45	LSR	4	1	Logical Shift Right
46	NEG	1	1	Negate
47	NOP	4	0	No Operation
48	ORA	2	2	OR Accumulators
49	PSH	4	1	Push Data
50	PUL	4	1	Pull Data
51	ROL	4	1	Rotate Left
52	ROR	4	1	Rotate Right
53	RTI	8	0	Return From Interrupt
54	RTS	8	0	Return From Subroutine
55	SBA	4	0	Subtract Acumulators
56	SBC	2	2	Subtract With Carry
57	SEC	4	0	Set Carry
58	SEI	4	0	Set Interrupt Mask
59	SEV	4	0	Set Overflow
60	STA	1	2	Store Accumulator
61	STS	1	1	Store Stack Register
62	STX	1	1	Store Index Register

TABLE D-1 (Continued)

6800 PARSER TABLE

	ASSM-INSTS	ASSM-CODES	ASSM-MODES	Description
63	SUB	2	2	Subtract
64	SWI	4	0	Software Interrupt
65	TAB	4	0	Transfer Accumulators
66	TAP	4	0	Transfer Accumulator to Condition Code Register
67	TBA	4	0	Transfer Accumulators
68	TPA	4	0	Transfer Condition Code Register to Accumulator
69	TST	2	0	Test
70	TXS	4	0	Transfer Stack Pointer to Index Register
71	TSX	4	0	Transfer Index Register to Stack Pointer
72	WAI	4	0	Wait For Interrupt
73	END	14	1	Define end of Source Program
74	EQU	13	1	Equate Symbol to an Operand
75	FCB	12	1	Form Constant Byte
76	FCC	12	1	Form Constant Characters
77	FDB	12	1	Form Double Constant Byte
78	NAM	14	1	Specify Name of Title
79	OPT	14	1	Control Assembler Options
80	ORG	14	1	Assign Origin of Program Counter
81	PAG	14	1	Skip to Next Page
82	RMB	11	1	Reserve Memory Bytes
83	SPC	14	1	Insert Space(s) in Output File

D.2 M68000

1. PARSER TABLE: The following describes the parser table (PM68000.TBL) used to analyze M68000 Programs:

```
CHARACTER*8   DELOPER(16)
CHARACTER*15  INSTRUMENT_VALUE
INTEGER       N_USER, N_ASSM, N_OPER
INTEGER*4     ASSM_CODES(200), ASSM_MODES(200)
INTEGER*4     ASSM_SPECS(200), USER_CODES(100)
INTEGER*4     USER_MODES(100), USER_SPECS(100)
INTEGER*4     PARSER_PRNT, COMMENT_PRNT, STRINGS_PRNT,
                INSTRUM_PRNT

 NAMELIST/PARSER_TABLE/INSTRUMENT_VALUE
*                     ,PARSER_PRNT
*                     ,STRINGS_PRNT
*                     ,COMMENT_PRNT
*                     ,INSTRM_PRNT
*                     ,N_ASSM
*                     ,ASSM_INSTS
*                     ,ASSM_CODES
*                     ,ASSM_MODES
*                     ,ASSM_SPECS
*                     ,N_USER
*                     ,USER_INSTS
*                     ,USER_CODES
*                     ,USER_MODES
*                     ,DELOPER
*                     ,USER_SPECS
```

Where:

N_ASSM, ASSM_INSTS, ASSM_CODE and ASSM_MODE are defined as in the 6800 description.

ASSM_SPECS = Special information about an instruction.

0 = No special information.

2 or 9 = Are ASSM_CODES which will override the general case ASSM_CODE associated with the particular instruction. This will happen autcmatically when the parser detects the instruction being used in a way that does not fit the general, default, case preset in the ASSM_MODES and ASSM_CODES.

USER_INSTS, USER_CODES, USER_MODES and USER_SPECS are used the same as the "ASSM" counterparts but for the user defined instructions such as macros. These are generated dynamically.

DELOPER = Delimiting operators, those not included in the "ASSM_INSTS" or "USER_INSTS", such as arithmetic operators (used for Halstead's calculations).

N_OPER = Number of operators in the "DELOPER" list.

2. M68000 NON-APPLICABLE METRICS: Also contained in the "Parser Table" is a namelist which contains any metrics which are predetermined to be non-applicable. The format is as follows:

```
      NAMELIST/INIT_MET/ N_COMMENT_LINES
     *                  ,N_LINES_X
     *                  ,N_LINES_NX
     *                  ,N_LINES
     *                  ,N_COMMENTS_X
     *                  ,N_COMMENTS_NX
     *                  ,N_INLINE_COMMENTS
     *                  ,N_PER_COMM_X
     *                  ,N_PER_COMM_NX
     *                  ,MAX_WO_COMMENTS
     *                  ,N_STAMNTS_X
     *                  ,N_STAMNTS_NX
     *                  ,MULT_LINE_STAMNTS
     *                  ,MULT_STAMNT_LINES
     *                  ,N_ENTRY_PTS
     *                  ,N_BRANCH_FRWD
     *                  ,N_BRANCH_BACK
     *                  ,N_BRANCH_OUT
     *                  ,N_BRANCH_COND
     *                  ,N_BRANCH_UNCOND
     *                  ,N_NODES
     *                  ,N_PATHS
     *                  ,MCCABES_CYCLOMATIC
     *                  ,N_LINES_SKIPPED
     *                  ,N_INCLUDES
     *                  ,N_OPERATORS
     *                  ,N_OPERATORS_USED
     *                  ,N_OPERANDS
     *                  ,N_OPERANDS_USED
     *                  ,N_PRIM_LINES
     *                  ,N_EMBED_LINES
     *                  ,N_SWITCH
```

```
*                    ,N_VARS
*                    ,CODE_COUNTS
```

This entry in the parser table should contain ONLY the metrics which are not applicable. For the M68000 language the following are set to -1 indicating non-applicability:

```
N_COMMENTS_X
N_COMMENTS_NX
N_PER_COMM_X
N_PER_COMM_NX
MULT_LINE_STAMNTS
MULT_STAMNTS_LINES
N_EMBED_LINES
N_SWITCH
```

3. M68000 LANGUAGE ANALYSIS LIMITATIONS:

Variable usage data relating more to higher level language, such as variable type and number of time a variable is passed as an argument, is not collected.

Table D-2 show the content of the Parser Table for the M68000 language.

TABLE D-2

M68000 PARSER TABLE

ASSM-INSTS	ASSM-MODES	ASSM-CODES	ASSM-SPECS	Description
ABCD	2	4	0	Add Decimal With Extend
ADD	2	4	0	Add Binary
ADDA	2	4	0	Add Address
ADDI	2	4	0	Add Immediate
ADDQ	2	4	0	Add Quick
ADDX	2	4	0	Add Extended
AND	2	4	0	AND Logical
ANDI	2	4	0	ADD Immediate
ASL	2	4	0	Arithmetic Shift
ASR	2	4	0	Arithmetic Shift
BCC	1	5	0	Branch Conditionally
BCS	1	5	0	Branch Conditionally
BEQ	1	5	0	Branch Conditionally
BGE	1	5	0	Branch Conditionally
BGT	1	5	0	Branch Conditionally
BHI	1	5	0	Branch Conditionally
BLE	1	5	0	Branch Conditionally
BLS	1	5	0	Branch Conditionally
BLT	1	5	0	Branch Conditionally
BMI	1	5	0	Branch Conditionally
BNE	1	5	0	Branch Conditionally
BPL	1	5	0	Branch Conditionally
BVC	1	5	0	Test a Bit
BVS	1	5	0	Test a Bit
BCHG	2	4	0	Test a Bit and Change
BCLR	2	4	0	Test a Bit and Clear
BRA	1	6	0	Branch Always
BSET	2	4	0	Test a Bit and Set
BSR	1	7	0	Branch to Subroutine
BTST	2	4	2	Test a Bit
CHK	2	5	0	Check Register Against Bounds

TABLE D-2 (Continued)

M68000 PARSER TABLE

ASSM-INSTS	ASSM-MODES	ASSM-CODES	ASSM-SPECS	Description
CLR	2	4	9	Clear an Operand
CMP	2	4	0	Compare
CMPA	2	4	0	Compare Address
CMPI	2	4	2	Compare Immediate
CMPM	2	4	0	Compare Memory
DBCC	2	5	0	Test Condition, Decrement, and Branch
DBCS	2	5	0	
DBEQ	2	5	0	
DBF	2	5	0	
DBGE	2	5	0	
DBGT	2	5	0	
DBHI	2	5	0	
DBLE	2	5	0	
DBLS	2	5	0	
DBLT	2	5	0	
DBMI	2	5	0	
DBNE	2	5	0	
DBPL	2	5	0	
DBT	2	5	0	
DBVC	2	5	0	
DBVS	2	5	0	
DBRA	2	6	0	
DIVS	2	4	0	Signed Divide
DIVU	2	4	0	Unsigned Divide
EOR	2	4	0	Exclusive OR Logical
EORI	2	4	0	Exclusive OR Immediate
EXG	2	4	0	Exchange Registers
EXT	1	4	0	Sign Extend

TABLE D-2 (Continued)
M68000 PARSER TABLE

ASSM-INSTS	ASSM-MODES	ASSM-CODES	ASSM-SPECS	Description
JMP	1	6	0	Jump
JSR	1	7	0	Jump to Subroutine
LEA	2	4	0	Load Effective Address
LINK	2	4	0	Link and Allocate
LSL	2	4	0	Logical Shift
LSR	2	4	0	Logical Shift
MOVE	2	4	9	Move Data From Source To Destination
MOVEA	2	4	U	Move Address
MOVEM	2	4	9	Move Miltiple Registers
MOVEP	2	4	0	Move Peripheral Data
MOVEQ	2	4	0	Move Quick
MULS	2	4	0	Signed Multiply
MULU	2	4	0	Unsigned Mulitply
NBCD	1	1	0	Negate Decimal With Extend
NEG	1	1	0	Negate
NEGX	1	1	0	Negate With Extend
NOP	0	4	0	No Operation
NOT	1	1	0	Logical Complement
OR	2	4	9	Inclusive OR Logical
ORI	2	4	9	Inclusive OR Immediate
PEA	1	2	0	Push Effective Address
RESET	0	4	0	Reset External Devices (Privileged Instruction)
ROL	2	4	0	Rotate (Without Extend)
ROR	2	4	0	Rotate (Without Extend)
ROXL	2	4	0	Rotate With Extend
ROXR	2	4	0	Rotate With Extend
RTE	0	8	0	Return From Exception (Privileged Instruction)
RTR	0	8	0	Return and Restore Condition Codes

TABLE D-2 (Continued)

M68000 PARSER TABLE

ASSM-INSTS	ASSM-MODES	ASSM-CODES	ASSM-SPECS	Description
RTS	0	8	0	Return From Subroutine
SBCD	2	4	0	Subtract Decimal With Extend
SCC	1	9	0	Set According to Condition
SCS	1	9	0	
SEQ	1	9	0	
SF	1	9	0	
SGE	1	9	0	
SGT	1	9	0	
SHI	1	9	0	
SLE	1	9	0	
SLS	1	9	0	
SLT	1	9	0	
SMI	1	9	0	
SNE	1	9	0	
SPI	1	9	0	
ST	1	9	0	
SVC	1	9	0	
SVS	1	9	0	
STOP	1	4	0	Load Status Register and Stop (Privileged Instruction)
SUB	2	4	0	Subtract Binary
SUBA	2	4	0	Subtract Address
SUBI	2	4	0	Subtract Immediate
SUBQ	2	4	0	Subtract Quick
SUBX	2	4	0	Subtract With Extendd
SWAP	1	4	0	Swap Register Halves
TAS	1	1	0	Test and Set an Operand
TRAP	1	6	0	Trap
TRAPV	0	5	0	Trap on Overflow
TST	1	4	0	Test an Operand
UNLK	1	4	0	Unlink

TABLE D-2 (Continued)

M68000 PARSER TABLE

ASSM-INSTS	ASSM-MODES	ASSM-CODES	ASSM-SPECS	Description
END	1	14	0	Program End
EQU	1	13	0	Assign Permanent Value
INCLUDE	1	14	0	Include Second File
OFFSET	1	14	0	Define Offsets
ORG	1	14	0	Absolute Origin
SECTION	1	14	0	Relocatable Prog Section
REG	1	14	0	Define Register List
SET	1	11	0	Assign Temporary Value
INFEQ	2	14	0	Conditional Assembly Decision
IFNE	2	14	0	Conditional Assembly Decision
IFLT	2	14	0	Conditional Assembly Decision
IFLE	2	14	0	Conditional Assembly Decision
IFLE	2	14	0	Conditional Assembly Decision
IFGT	2	14	0	Conditional Assembly Decision
IFGE	2	14	0	Conditional Assembly Decision
ENDC	0	14	0	End Conditional Assembly
COMLINE	1	12	0	Command Line
DC	1	13	0	Define Constants
DCB	2	13	0	Define Constant Block
DS	1	12	0	Define Storage
FAIL	1	14	0	Programmer Generated Error
FORMAT	0	14	0	Enable Auto Formatting
NOFORMAT	0	14	0	Disable Auto Formatting
LIST	0	14	0	Enable Listing
NOL	0	14	0	Disable Listing
NOLIST	0	14	0	Disable Listing
LLEN	1	14	0	Set Line Lengths
NOOBJ	0	14	0	Disable Object Output
OPT	0	14	0	Assembler Options
PAGE	0	14	0	Top of Page
NOPAGE	0	14	0	Disable Paging
SPC	1	14	0	Skip n Lines

TABLE D-2 (Continued)

M68000 PARSER TABLE

ASSM-INSTS	ASSM-MODES	ASSM-CODES	ASSM-SPECS	Description
TTL	1	14	0	Title
IDNT	0	14	0	Relocatable ID Record
XDEF	1	11	0	External Symbol Def
XREF	1	11	0	External Symbol Ref
IF	0	3	0	Structured Constructs
ELSE	0	3	0	Structured Constructs
ENDI	0	14	0	Structured Constructs
FOR	0	3	0	Structured Constructs
ENDF	0	14	0	Structured Constructs
REPEAT	0	14	0	Structured Constructs
UNTIL	0	3	0	Structured Constructs
WHILE	0	3	0	Structured Constructs
ENDW	0	14	0	Structured Constructs

D.3 SKC

1. PARSER TABLE: The following describes the parser table (PSKC.TBL) used to analyze SKC Programs:

```
CHARACTER*8   DELOPER(16)
CHARACTER*15  INSTRUMENT_VALUE
INTEGER       N_USER, N_ASSM, N_OPER
INTEGER*4     ASSM_CODES(250), ASSM_MODES(250)
INTEGER*4     ASSM_SPECS(250), USER_CODES (100)
INTEGER*4     USER_MODES(100), USER_SPECS(100)
INTEGER*1     PARSER_PRNT, COMMENT_PRNT, STRINGS_PRNT,
                INSTRUM_PRNT

 NAMELIST/PARSER_TABLE/INSTRUMENT_VALUE
*                     ,PARSER_PRNT
*                     ,STRINGS_PRNT
*                     ,COMMENT_PRNT
*                     ,INSTRM_PRNT
*                     ,N_ASSM
*                     ,ASSM_INSTS
*                     ,ASSM_CODES
*                     ,ASSM_MODES
*                     ,ASSM_SPECS
*                     ,USER_CODES
*                     ,USER_MODES
*                     ,USER_SPECS
*                     ,DELOPER
*                     ,N_OPER
```

Where:

N_ASSM, ASSM_INSTS, ASSM_CODE and ASSM_MODE are defined as in the 6800 description.

ASSM_SPECS = Special information about an instruction (not used for SKC). USER_INSTS, USER_CODES, USER_MODES and USER_SPECS are used the same as the "ASSM" counterparts but for the user defined instructions such as macros. These are generated dynamically.

DELOPER = Delimiting operators, those not included in the "ASSM_INSTS" or "USER_INSTS", such as arithmetic operators (used for Halstead's calculations).

N_OPER = Number of operators in the "DELOPER" list.

2. SKC NON-APPLICABLE METRICS: Also contained in the "Parser Table" is a namelist which contains any metrics which are predetermined to be non-applicable. The format is as follows:

```
      NAMELIST/INIT_MET/ N_COMMENT_LINES
     *                  ,N_LINES_X
     *                  ,N_LINES_NX
     *                  ,N_LINES
     *                  ,N_COMMENTS_X
     *                  ,N_COMMENTS_NX
     *                  ,N_INLINE_COMMENTS
     *                  ,N_PER_COMM_X
     *                  ,N_PER_COMM_NX
     *                  ,MAX_WO_COMMENTS
     *                  ,N_STAMNTS_X
     *                  ,N_STAMNTS_NX
     *                  ,MULT_LINE_STAMNTS
     *                  ,MULT_STAMNT_LINES
     *                  ,N_ENTRY_PTS
     *                  ,N_BRANCH_FRWD
     *                  ,N_BRANCH_BACK
     *                  ,N_BRANCH_OUT
     *                  ,N_BRANCH_COND
     *                  ,N_BRANCH_UNCOND
     *                  ,N_NODES
     *                  ,N_PATHS
     *                  ,MCCABES_CYCLOMATIC
     *                  ,N_LINES_SKIPPED
     *                  ,N_INCLUDES
     *                  ,N_OPERATORS
     *                  ,N_OPERATORS_USED
     *                  ,N_OPERANDS
     *                  ,N_OPERANDS_USED
     *                  ,N_PRIM_LINES
     *                  ,N_EMBED_LINES
     *                  ,N_SWITCH
     *                  ,N_VARS
     *                  ,CODE_COUNTS
```

This entry in the parser table should contain ONLY the metrics which are not applicable. For the SKC language the following are set to -1 indicating non-applicability:

```
N_COMMENTS_X
N_COMMENTS_NX
N_PER_COMM_X
N_PER_COMM_NX
N_INCLUDES
MULT_LINE_STAMNTS
```

MULT_STAMNTS_LINES
N_EMBED_LINES
N_SWITCH

3. SKC LANGUAGE ANALYSIS LIMITATIONS:

a. Variable usage data relating more to higher level language, such as variable type and number of time a variable is passed as an argument, is not collected.

b. Metrics for lines which make up a Macro Definition are not collected. However, Macro Definition and Usage Information can be found in the Macro Report.

Table D-3 show the content of the Parser Table for the SKC language.

TABLE D-3

SKC PARSER TABLE

ASSM INSTS	C	M	DESCRIPTION
ADD16	2	2	Add 16 bits.
ADD32	2	2	Add 32 bits.
ADDR	2	2	Add general register.
ADF	2	2	Add floating.
AFD	2	2	Add floating DP.
AND16	2	2	Logical AND 16 bits.
AND32	2	2	Logical AND 32 bits.
CBIT	2	2	Compare bit in memory.
COM16	2	2	Compare 16 bits.
COM32	2	2	Compare 32 bits.
COMF	2	2	Compare floating.
COMFD	2	2	Compare floating DP.
OR	2	2	Compare general register.
DVD16	2	2	Divide 16 bits.
DVDX16	2	2	Divide extended precision 16 bits.
DVD32	2	2	Divide 32 bits.
DVDX32	2	2	Divide extended precision 32 bits.
DVF	2	2	Divide floating.
EXEC	2	1	Execute.
EXO16	2	2	Exclusive OR 16 bits.
EXO32	2	2	Exclusive OR 32 bits.
IDVF	2	2	Inverse divide floating.
ISBF	2	2	Inverse subtract floating.
ISFD	2	2	Inverse subtract floating DP.
LAE	4	1	Load AU with ffctive address.
LD16	2	2	Load 16 bits.
LD32	2	2	Load 32 bits.
LD64	2	2	Load 64 bits.
LDMK	2	2	Load mask register (not available on SKC3121).
LDPUSH	2	2	Load 32 bits and push.
LDR	2	2	Load general register.
LMMR	2	2	Load memory management RAM (not available on SKC3121).

TABLE D-3 (Continued)

SKC PARSER TABLE

ASSM INSTS	C	M	DESCRIPTION
MLF	2	2	Multiply floating.
MLFT	2	2	Multiply floating truncated.
MUL16	2	2	Multiply 16 bits.
MUL32	2	2	Multiply 32 bits.
MULR	2	2	Multiply general register.
OR16	2	2	Logical OR 16 bits.
OR32	2	2	Logical OR 32 bits.
RBIT	1	2	Reset bit in memory.
RDVF	2	2	Remainder divide floating.
RESTORE	2	2	Load block of registers.
RIDVF	2	2	Remainder inverse divide floating.
RTA	6	2	Return jump.
RTAP	6	2	Return from priority program interrupts (not available on SKC 3121).
SAVE	9	2	Store block of registers.
SBF	2	2	Subtract floating.
SBIT	1	2	Set bit in memory.
SFD	2	2	Subtract floating.
ST16	9	2	Store 16 bits.
ST32	9	2	Store 32 bits.
ST64	9	2	Store 64 bits.
STPOP	9	2	Store 32 bits and pop.
STR	9	2	Store general register.
SUB16	2	2	Subtract 16 bits.
SUB32	2	2	Subtract 32 bits.
SUBR	2	2	Subtract general register.
TBIT	1	2	Toggle bit in memory.
ABS16	4	0	Absolute value - 16 bit.
ABS32	4	0	Absolute value - 32 bit.
ABSF	4	0	Absolute value - floating point.
ABSFD	2	2	Absolute value - floating point.
CFX	4	0	Convert floating to fixed point.

TABLE D-3 (Continued)

SKC PARSER TABLE

ASSM INSTS	C	M	DESCRIPTION
CL16	4	0	Clear 16 bits.
CL32	4	0	Clear 32 bits.
CL32F	4	0	Clear 32 bits floating.
CL64F	4	0	Clear 64 bits floating.
CLB	4	0	Clear B register.
CLR	4	1	Clear general register.
CXF	4	0	Convert fixed to floating point.
DPI	4	0	Disable program interrupts.
EPI	4	0	Enable program interrupts.
EPPI	4	0	Enable priority program interrupts.
EXAB	4	0	Exchange A and B register.
EXTDV	4	0	Use extended precision numerator in following divide.
HALT	4	0	Halt the CPU.
LRA	4	1	Load register from AU register.
LAR	4	1	Load AU register from general register.
NEG16	4	0	Negate (two's complement) 16 bits.
NEG32	4	0	Negate (two's complement) 32 bits.
NEGF	4	0	Negate floating point.
NEGFD	4	0	Negate floating point double precision.
NEGR	4	1	Negate (two's complement) general register.
NOT16	4	0	Not (one's complement) 16 bits.
NOT32	4	0	Not (one's complement) 32 bits.
SQRT	4	0	Square-root-floating point.
SQUARE	4	0	Square-floating point.
ADDRR	4	2	Add register to register.
ANDRR	4	2	Logical AND register with register.
CORR	4	2	Compare register with register.
DVDRR	4	2	Divide register by register integer.
EXORR	4	2	Exclusive OR register with register.
EXRR	4	2	Exchange register with register.
LDRR	4	2	Load register to register.
LRPC	4	2	Load general register with PC.

TABLE D-3 (Continued)

SKC PARSER TABLE

ASSM INSTS	C	M	DESCRIPTION
MULRR	4	2	Multiply register by register integer.
ORRR	4	2	Logical OR register with register.
SLCR	4	2	Shift left circular register.
SLLR	4	2	Shift left logical register.
SRAR	4	2	Shift right algebraic register.
SRLR	4	2	Shift right logical register.
SUBRR	4	2	Subtract register from register.
SLC	2	2	Shift left circular.
SLCD	2	2	Shift left circular double.
SLL	2	2	Shift left logical.
SLLD	2	2	Shift right logical double.
SRAD	2	2	Shift right algebraic double.
SRC	2	2	Shift right circular.
SRCD	2	2	Shift right circular double.
SRLD	2	2	Shift right logical double.
IN1	4	1	Input direct.
IN2	2	2	Input via memory.
OUT1	4	1	Output direct.
OUT2	2	2	Output via memory.
JEQ	5	2	Jump equal.
JGE	5	2	Jump greater than or equal.
JGT	5	2	Jump greater than.
JLE	5	2	Jump less than or equal.
JLT	5	2	Jump less than.
JNE	5	2	Jump not equal.
JSA	7	2	Subroutine jump-absolute.
JSAI	7	2	Subroutine jump-absolute indirect.
JSAIR	7	2	Subroutine jump-absolute indirect relative.
JSR	7	2	Subroutine jump-relative.
JSRI	7	2	Subroutine jump-relative indirect.
JU	6	2	Jump unconditional.
JUA	6	2	Jump unconditional absolute.

TABLE D-3 (Continued)

SKC PARSER TABLE

ASSM INSTS	C	M	DESCRIPTION
JUAI	6	2	Jump unconditional absolute indirect.
JUAIR	6	2	Jump unconditional absolute indirect relative.
JURI	6	2	Jump relative indirect.
ADFS	4	0	Add stack floating point.
COMFS	4	0	Compare stack floating point.
DVFS	4	0	Divide stack floating point.
IDVFS	4	0	Inverse divide stack floating point.
ISBFS	4	0	Inverse subract stack floating point.
MLFS	4	0	Multiply stack floating point.
POP	4	0	Pop the stack.
PUSH	4	0	Push the stack.
SBFS	4	0	Subtract stack floating point.
PROL	2	2	Fixed prologue.
PROLS	2	2	Fixed prologure and save.
PROLV	2	1	Variable prologue.
PROLVS	2	1	Variable prologue and save.
RETURN	8	0	Return from subroutine.
BMOVE	4	1	Block move.
TRAPX16	2	2	Trap 16 bits fixed point.
TRAPX32	2	2	Trap 32 bits fixed point.
TRAPF32	2	2	Trap 32 bits floating point.
TRAPF64	2	2	Trap 64 bits floating point.
DVFD	2	2	Divide floating DP.
IDVFD	2	2	Inverse divide floating DP.
MLFD	2	2	Multiply floating DP.
NOP	4	0	No operation.
ADRS	13	1	ADdReSs.
VFD	11	2	Variabl field definition.
ASCII	12	1	ASCII characters.
ASCIIP	12	1	ASCII characters.
EQU	13	1	EQUate.
SETD	13	1	Define SET symbol-Decimal.

TABLE D-3 (Continued)

SKC PARSER TABLE

ASSM INSTS	C	M	DESCRIPTION
SETX	13	1	Define SET symbol-heX.
BIT	13	2	Define BIT symbol.
BASE	14	1	Activate BASE register.
DBASE	14	1	Deactivate BASE register.
UBASE	14	1	Unconditional BASE register.
ENTRY	14	1	ENTRY point list.
GDATA	14	0	Global DATA list.
MAIN	14	0	MAIN deck.
END	14	0	END of deck.
INT	14	0	INTerrupt deck.
LIST	14	0	Source LISTing.
USE	14	1	USE location counter.
ORG	14	0	Absolute ORiGin.
EVEN	14	0	EVEN location.
COMMON	14	0	COMMON data region.
BSTACK	11	1	Block STACK.
DEC16	13	1	DECimal data (16 bits).
DEC32	13	1	DECimal data (32 bits).
DEC64	13	1	DECimal data (64 bits).
HEX16	13	1	HEXadecimal data (16 bits).
HEX32	13	1	HEXadecimal data (32 bits).
HEX64	13	1	HEXadecimal data (64 bits).
SCLB16	13	2	SCaLed Binary data (16 bits).
SCLB32	13	2	SCaLed Binary data (32 bits).
SCLW16	13	2	SCaLed Weighted data (16 bits).
SCLW32	13	2	SCaLed Weighted data (32 bits).
BSS	11	1	Block Started by Symbol.
BES	11	1	Block Ended by Symbol.
UNLIST	14	0	UNdo source.
TTL	14	0	TiTLe.
EJECT	14	0	EJECT rest of page.
SPACE	14	0	SPACE by d lines.

TABLE D-3 (Continued)

SKC PARSER TABLE

ASSM INSTS	C	M	DESCRIPTION
CSB	14	0	Compool Symbols Begin.
CPOOL	14	1	Declare ComPOOL symbol.
CSE	14	0	Compool Symbols End.
PTR16	2	2	PoinTeR 16.
PTR32	2	2	PoinTeR 32.
PTR64	2	2	PoinTeR 64.
SBITM	1	2	Set BIT Marco.
RBITM	1	2	Reset BIT Marco.
TBITM	1	2	Toggle BIT Marco.
CBITM	2	2	Compare BIT Macro.
JBIT1	5	2	Jump if BIT is 1.
JBIT0	5	2	Jump if BIT is 0.
ATI	14	0	Memory Access Time of Instruction.
ATO	14	0	Memory Access Time of Operand.
ATIME			Execution Timing weight.
CTI	14	0	Memory Cycle Time of Instruction.
CTO	14	0	Memory Cycle Time of Operand.

D.4 PDP11

1. PARSER TABLE: The following describes the parser table (PDP11.TBL) used to analyze PDP11 Programs:

```
CHARACTER*8   DELOPER(16)
CHARACTER*15  INSTRUMENT_VALUE
INTEGER       N_USER, N_ASSM, N_OPER
INTEGER*4     ASSM_CODES(200), ASSM_MODES(200)
INTEGER*4     ASSM_SPECS(200), USER_CODES(100)
INTEGER*4     USER_MOFRD(100), USER_SPECS(100)
INTEGER*4     PARSER_PRNT, COMMENT_PRNT, STRINGS_PRNT,
                INSTRUM_PRNT

 NAMELIST/PARSER_TABLE/INSTRUMENT_VALUE
*                     ,PARSER_PRNT
*                     ,STRINGS_PRNT
*                     ,COMMENT_PRNT
*                     ,INSTRM_PRNT
*                     ,N_ASSM
*                     ,ASSM_INSTS
*                     ,ASSM_CODES
*                     ,ASSM_MODES
*                     ,ASSM_SPECS
*                     ,N_USER
*                     ,USER_INSTS
*                     ,USER_CODES
*                     ,USER_MODES
*                     ,DELOPER
*                     ,USER_SPECS
```

Where:

N_ASSM, ASSM_INSTS, ASSM_CODE and ASSM_MODE are defined as in the 6800 description.

USER_INSTS, USER_CODES, and USER_MODES are used the same as the "ASSM" counterparts but for the user defined instructions such as macros. These are generated dynamically.

DELOPER = Delimiting operators, those not included in the "ASSM_INSTS" or "USER_INSTS", such as arithmetic operators (used for Halstead's calculations).

N_OPER = Number of operators in the "DELOPER" list.

2. PDP11 NON-APPLICABLE METRICS: Also contained in the "Parser Table" is a namelist which contains any metrics which are predetermined to be non-applicable. The format is as follows:

```
      NAMELIST/INIT_MET/ N_COMMENT_LINES
     *                  ,N_LINES_X
     *                  ,N_LINES_NX
     *                  ,N_LINES
     *                  ,N_COMMENTS_X
     *                  ,N_COMMENTS_NX
     *                  ,N_INLINE_COMMENTS
     *                  ,N_PER_COMM_X
     *                  ,N_PER_COMM_NX
     *                  ,MAX_WO_COMMENTS
     *                  ,N_STAMNTS_X
     *                  ,N_STAMNTS_NX
     *                  ,MULT_LINE_STAMNTS
     *                  ,MULT_STAMNT_LINES
     *                  ,N_ENTRY_PTS
     *                  ,N_BRANCH_FRWD
     *                  ,N_BRANCH_BACK
     *                  ,N_BRANCH_OUT
     *                  ,N_BRANCH_COND
     *                  ,N_BRANCH_UNCOND
     *                  ,N_NODES
     *                  ,N_PATHS
     *                  ,MCCABES_CYCLOMATIC
     *                  ,N_LINES_SKIPPED
     *                  ,N_INCLUDES
     *                  ,N_OPERATORS
     *                  ,N_OPERATORS_USED
     *                  ,N_OPERANDS
     *                  ,N_OPERANDS_USED
     *                  ,N_PRIM_LINES
     *                  ,N_EMBED_LINES
     *                  ,N_SWITCH
     *                  ,N_VARS
     *                  ,CODE_COUNTS
```

This entry in the parser table should contain ONLY the metrics which are not applicable. For the PDP11 language the following are set to -1 indicating non-applicability:

```
N_COMMENTS_X
N_COMMENTS_NX
N_PER_COMM_X
```

N_PER_COMM_NX
MULT_LINE_STAMNTS
MULT_STAMNTS_LINES
N_EMBED_LINES
N_SWITCH

3. PDP11 LANGUAGE ANALYSIS LIMITATIONS: Variable usage data relating more to higher level language, such as variable type and number of time a variable is passed as an argument, is not collected.

Data on nested macro definitions is not collected. Table D-4 shows the content of the Parser Table for the PDP11 language.

TABLE D-4

PDP11 PARSER TABLE

ASSM-INSTS	ASSM-MODES	ASSM-CODES	Description
ADC	4	1	Add Carry
ADCB	4	1	Add Carry (Byte)
ADD	4	2	Add Source to Destination
ASH	4	2	Shift Arithmetically
ASHC	4	2	Arithmethic Shift Combined
ASL	4	1	Arithmetic Shift Left
ASLB	4	1	Arithmetic Shift Left (Byte)
ASR	4	1	Arithmetic Shift Right
ASRB	4	1	Arithmetic Shift Right (Byte)
BCC	5	1	Branch if Carry is Clear
BCS	5	1	Branch if Carry is Set
BEQ	5	1	Branch if Equal
BHE	5	1	Branch if Greater Than or Equal
BGT	5	1	Branch if Greater Than
BHI	5	1	Branch if Higher
BHIS	5	1	Branch if Higher or Same
BIC	4	2	Bit Clear
BICB	4	2	Bit Clear (Byte)
BIS	4	2	Bit Set
BISB	4	2	Bit Set (Byte)
BIT	3	2	Bit Test
BITB	3	2	Bit Test (Byte)
BLE	5	1	Branch if Less Than or Equal
BLO	5	1	Branch if Lower
BLOS	5	1	Branch if Lower or Same
BLT	5	1	Branch if Less Than
BMI	5	1	Branch if Minus
BNE	5	1	Branch if Not Equal
BPL	5	1	Branch if Plus
BPT	6	0	Breakpoint Trap
BR	6	1	Branch Unconditional
BVC	5	1	Branch if Overflow is Clear

TABLE D-4 (Continued)

PDP11 PARSER TABLE

ASSM-INSTS	ASSM-MODES	ASSM-CODES	Description
BVS	5	1	Branch is Overflow is Set
CALL	7	1	Jump to Subroutine (JSR PC, xxx)
CCC	4	0	Clear All Condition Codes
CLC	4	0	Clear C Condition Code Bit
CLN	4	0	Clear N Condition Code Bit
CLR	4	1	Clear Destination
CLRB	4	1	Clear Destination (Byte)
CLV	4	0	Clear V Condition Code Bit
CLZ	4	0	Clear Z Condition Code Bit
CMP	3	2	Compare Source to Destination
CMPB	3	1	Complement Destination
COM	4	1	Complement Destination
COMB	4	1	Complement Destination (Byte)
DEC	1	1	Decrement Destination
DECB	1	1	Decrement Destination (Byte)
DIV	4	2	Divide
EMT	6	0	Emulator Trap
FADD	4	2	Floating Add
FDIV	4	2	Floating Divide
FMUL	4	2	Floating Multiply
FSUB	4	2	Floating Subtract
HALT	4	0	Halt
INC	1	1	Increment Destination
INCB	1	1	Increment Destination (Byte)
IOT	6	0	Input/Output Trap
JMP	6	1	Jump
JSR	7	2	Jump to Subroutine
MARK	4	1	Mark
MFPI	4	1	Move From Previous Instruction Space
MFPS	4	1	Move From PS (LSI-11)
MOV	4	2	Move Source to Destination
MOVB	4	2	Move Source to Destination (Byte)

TABLE D-4 (Continued)

PDP11 PARSER TABLE

ASSM-INSTS	ASSM-MODES	ASSM-CODES	Description
MTPI	4	1	Move to Previous Instruction Space
MTPS	4	1	Move to PS(LSI-11)
MUL	4	2	Multiply
NEG	1	1	Negate Destination
NEGB	1	1	Negate Destination (Byte)
NOP	4	1	No Operation
RESET	4	0	Reset External Bus
RETURN	8	0	Return From Subroutine (RTS PC)
ROL	4	1	Rotate Left
ROLB	4	1	Rotate Left (Byte)
ROR	4	1	Rotate Right
RORB	4	1	Rotate Right (Byte)
RTI	8	0	Return From Interrupt (Permits a Trace Trap)
RTS	8	1	Return From Subroutine
RTT	8	0	Return From Interrupt (Inhibits Trace Trap)
SBC	4	1	Subtract Carry
SBCB	4	1	Subtract Carry (Byte)
SCC	1	0	Set All Condition Code Bits
SEC	1	0	Set C Condition Code Bit
SEN	1	0	Set N Condition Code Bit
SEV	1	0	Set V Condition Code Bit
SEZ	1	0	Set Z Condition Code Bit
SOB	5	2	Subtract One And Branch
SUB	4	2	Subtract Source From Destination
SWAB	4	1	Swap Bytes
SXT	4	1	Sign Extend
TRAP	6	1	Trap
TST	3	1	Test Destination
TSTB	3	1	Test Destination (Byte)
WAIT	4	0	Wait For Interrupt

TABLE D-4 (Continued)

PDP11 PARSER TABLE

ASSM-INSTS	ASSM-MODES	ASSM-CODES	Description
XOR	4	2	Exclusive OR
.ASCII	14	0	Translates character string to ASCII equivlents.
.ASCIZ	14	0	Translates character string to ASCII equivalents; inserts zero byte as last character.
.ASECT	14	0	Begins absolute program section (provided for compatibility with other PDP11 assembliers).
.BLKB	14	0	Reserves byte block in accordance with value of specified argument.
.BLKW	14	0	Reserves word block in accordance with value of specified argument.
.BYTE	14	0	Generates successive byte data in accordance with specified arguments.
.CSECT	14	0	Begins relocatable program section (provided for compatibility with other PDP11 assemblers).
.DSABL	14	0	Disables speified function.
.ENABL	14	0	Enables specified function.
.END	14	0	Defines logical end of source program.
.ENDC	14	0	Defines end of conditional assembly block.
.ENDR	14	0	Defines end of current repeat block (provided for compatibility with other PDP11 assemblers).
.EOT	14	0	Define End of Tap Condition (ignored).
.ERROR	14	0	Outputs diagnostic message to listing filo or command output device.
.FLT2	14	0	Causes two words of storage to be generated for each floating-point argument.

TABLE D-4 (Continued)

PDP11 PARSER TABLE

ASSM-INSTS	ASSM-MODES	ASSM-CODES	Description
.FLT4			Causes four words of storage to be generated for each floating-point argument.
.GLOBL	14	0	Declares global attribute for specified symbol(s).
.IDENT	14	0	Labels object module with specified program version number.
.IF	14	0	Begins conditional assembly block.
.IFF	14	0	Begins subconditional assembly block (if conditional assembly block test is false).
.IFT	14	0	Begins subconditional assembly block (if conditional assembly block test is true).
.IFTF	14	0	Begins subconditional assembly block (whether conditional assembly block test is true or false).
.IIF	14	0	Assembles immediate conditional assembly statement (if specified condition is satisfied).
.IRP	14	0	Begins indefinite repeat block; replaces specified symbol with specified successive real arguments.
.IRPC	14	0	Begins indefinite repear block; replaces specified symbol with value of successive characters in specified string.
.LIMIT	14	0	Reserves two words of storage for high and low addresses of task image.
.LIST	14	0	Controls listing level count and format of assembly listing. MACRO denotes start of macro definition.
.MCALL	14	0	Identifies required macro definition(s) for assembly.
.MEXIT	14	0	Exit from current macro definition or indefinite repeat block.

TABLE D-4 (Continued)

PDP11 PARSER TABLE

ASSM-INSTS	ASSM-MODES	ASSM-CODES	Description
.NARG	14	0	Equates specified symbol to the number of arguments in the macro expansion.
.NCHR	14	0	Equates specified symbol to the number of characters in the specified character string.
.NLIST	14	0	Controls listing level count and suppresses specified portions of the assembly listing.
.NTYPE	14	0	Equates specified symbols to the addressing mode of the specified argument.
.ODD	14	0	Byte-aligns the current location counter.
.PAGE	14	0	Advances form to top of next page.
.PRINT	14	0	Prints specified message on command output deivce.
.PSECT	14	0	Begins specified program section having specified attributes.
.RADIX	14	0	Changes current program radix to specified radix.
.RADSO	14	0	Generates data block having Radix-50 equivalents of specified character string.
.REPT	14	0	Begins repeat block and replicates it according to the value of the specjfied expression.
.SBTTL	14	0	Prints specified subtitle text as the second line of the assembly listing page header.
.TITLE	14	0	Prints specified title text as object module name in the first line of the assembly listing page header.
.WORD	14	0	Generats successive word data in accordance with specified arguments.
.EVEN	14	0	Word-aligns the current location counter.

(THIS PAGE INTENTIONALLY BLANK)

END

FILMED

[redacted]

DTIC